



TUGAS AKHIR - TM 141585

**KALIBRASI STEREO KAMERA DAN
PENENTUAN KOORDINAT 3 DIMENSI
UNTUK TARGET TUNGGAL PADA SISTEM
PELONTAR PELURU *AUTOTRACKING***

WARDAH CHOIRINA LUTFI
NRP 2111 100 025

Dosen Pembimbing:
Arif Wahjudi, ST, MT, PhD.

JURUSAN TEKNIK MESIN
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya
2016



FINAL PROJECT - TM 141585

**STEREO CAMERA CALIBRATION AND
DETERMINATION OF 3 DIMENSIONAL
COORDINATES FOR SINGLE TARGET ON
AUTOTRACKING BULLET LAUNCHER
SYSTEM**

WARDAH CHOIRINA LUTFI
NRP 2111 100 025

Academic Supervisor:
Arif Wahjudi, ST, MT, PhD.

DEPARTMENT OF MECHANICAL ENGINEERING
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya
2016



TUGAS AKHIR - TM 141585

**KALIBRASI STEREO KAMERA DAN
PENENTUAN KOORDINAT 3 DIMENSI
UNTUK TARGET TUNGGAL PADA SISTEM
PELONTAR PELURU *AUTOTRACKING***

WARDAH CHOIRINA LUTFI
NRP 2111 100 025

Dosen Pembimbing:
Arif Wahjudi, ST, MT, PhD.

JURUSAN TEKNIK MESIN
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

LEMBAR PENGESAHAN

KALIBRASI *STEREO* KAMERA DAN PENENTUAN KOORDINAT 3 DIMENSI UNTUK TARGET TUNGGAL PADA SISTEM PELONTAR PELURU *AUTOTRACKING*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat Memperoleh Gelar Sarjana Teknik
Mesin Pada Bidang Studi Manufaktur
Program Studi S-1
Jurusan Teknik Mesin
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember Surabaya

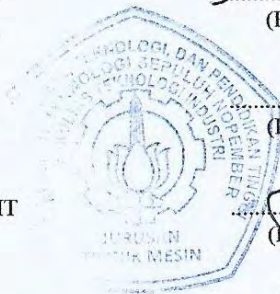
Oleh :

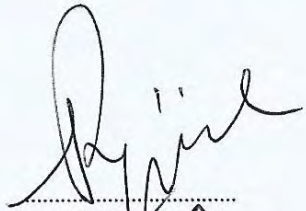
WARDAH CHOIRINA LUTFI

NRP : 2111 100 025

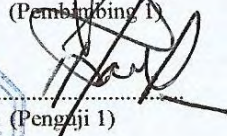
Disetujui oleh Tim Penguji Tugas Akhir

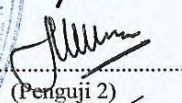
1. Arif Wahyudi, ST., MT., Ph.D
(NIP. 197303222001121001)
2. Ir. Sampurno, MT
(NIP 196504041989031002)
3. Ari Kurniawan Saputra, ST, MT
(NIP. 210150201)
4. Dinny Harnany, ST, MSc
(NIP. 2100201405001)

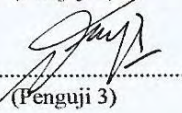




(Pembimbing 1)


(Penguji 1)


(Penguji 2)


(Penguji 3)

KALIBRASI STEREO KAMERA DAN PENENTUAN KOORDINAT 3 DIMENSI UNTUK TARGET TUNGGAL PADA SISTEM PELONTAR PELURU AUTOTRACKING

Nama : WARDAH C. LUTFI
NRP : 2111 100 025
Jurusan : TEKNIK MESIN FTI-ITS
Dosen Pembimbing : Arif Wahjudi, ST, MT, PhD.

ABSTRAK

Teknologi pengukuran semakin berkembang seiring dengan pesatnya perkembangan jaman. Banyak ditemukan teknologi pengukuran yang semakin canggih seperti non-contact measurement yang memanfaatkan gelombang ultrasonik, laser, bahkan kamera. Selain dapat digunakan sebagai alat pengukur jarak, kamera juga bisa digunakan untuk mendeteksi sebuah obyek dengan memanfaatkan teknologi pengolahan citra digital. Teknologi ini dapat dimanfaatkan pada banyak bidang, seperti misalnya bidang industri, bidang manufaktur, bahkan bidang militer. Teknologi ini mulai dikembangkan dibidang militer dalam bentuk senjata otomatis dikarenakan dapat mengurangi jumlah korban jiwa dari pihak militer. Saat ini sudah terdapat prototipe mesin penembak berupa alat pelontar peluru. Maka dari itu pada Tugas Akhir ini diharapkan tercipta program pendeteksi, pelacak dan penentu koordinat 3 dimensi yang dapat diaplikasikan pada prototipe tersebut sehingga alat dapat menembak obyek tertentu tanpa memasukkan nilai koordinat secara manual.

Langkah pengerjaan penelitian ini dengan melakukan proses kalibrasi stereo kamera menggunakan papan catur yang didapat dari library OpenCV. Dari proses tersebut nilai – nilai intrinsik dari tiap kamera didapat. Kemudian image processing dilakukan dengan input sebuah image. Image tersebut diolah

menggunakan software Visual Studio dan library OpenCV. Pada image processing ini image yang didapat diolah untuk mendeteksi obyek berdasarkan warna. Setelah obyek terdeteksi dilakukan proses untuk menentukan koordinat z atau jarak objek terhadap kedua kamera. Proses ini menggunakan data yang didapat dari proses kalibrasi yang berupa nilai intrinsik kamera dan dengan menggunakan suatu persamaan. Setelah didapat nilai koordinat z, nilai ini digunakan untuk mencari koordinat x dan y dari obyek.

Hasil tugas akhir ini adalah performa dari program pendeteksian objek tunggal dan diam secara real – time ini sebesar 100%. Sedangkan untuk program penentuan koordinat 3 dimensi objek diketahui bahwa terdapat error sebesar 0,0075%, 0,092% dan 0,085% untuk masing – masing nilai koordinat z atau jarak objek terhadap kamera dengan toleransi ± 6 cm, nilai koordinat x dan nilai koordinat y dengan toleransi ± 3 cm . Kesalahan terjadi sangat dipengaruhi oleh pencahayaan yang tidak merata saat dilakukan pengujian, kualitas kamera yang dipakai, serta perbedaan antara perubahan posisi objek yang ditangkap oleh kamera dan perubahan posisi objek yang sebenarnya.

Kata kunci : Image processing, Kalibrasi stereo kamera, Non contact measurement, Pelacakan objek, Pengukuran.

STEREO CAMERA CALIBRATION AND DETERMINATION OF 3 DIMENSIONAL COORDINATES FOR SINGLE TARGET ON AUTOTRACKING BULLET LAUNCHER SYSTEM

Name : WARDAH C. LUTFI
NRP : 2111 100 025
Department : MECHANICAL ENGINEERING
Academic Supervisor : Arif Wahjudi, ST, MT, PhD.

ABSTRACT

Measurement technology is growing in line with the rapid development of the times. There is a lot of measurement technologies that increasingly the measurement technology such as non-contact measurement that utilizes ultrasonic waves, lasers, and even cameras. In addition it can be used as a distance measuring device, the camera can also be used to detect an object by using image processing technology. This technology can be utilized in many fields, such as industrial, manufacturing, and even military fields. This technology was developed in the field of military in the form of automatic weapons so it can reduce the victim numbers of the military. There is a prototype machine gunner that called bullet launcher tool. Therefore in this final project is expected to create a program that can detecting, tracking and determining the 3 dimensional coordinates of an object that can be applied to the prototype. So the tool can shoot a particular object without manually entering coordinate values.

The step of this study is by doing the stereo camera calibration process using a chessboard that obtained from OpenCV library. The process value from that process are the intrinsic value of each camera. Then, the image processing with the input image. Image was processed using Visual Studio software and the OpenCV library. Image processing is processed to detect

objects by color. After the object detected, the next process is to determine the z coordinates of the object or the distance of the two cameras. This process uses data obtained from the calibration process that form of intrinsic value of the camera by using an equation. After obtained the z coordinate value, this value is used to find the x and y coordinates of the object.

From this thesis showed that the performance of a single object detection real - time program is 100%. As for determining the 3-dimensional coordinates of the object program is known that there is an error of 0.0075%, 0.092% and 0.085% for each z coordinate value or distance of an object to the camera with a tolerance of ± 6 cm, the value of the coordinates x and y coordinate values with tolerance ± 3 cm. Mistakes that might happen very influenced by uneven lighting when testing process, the quality of the camera that we used, and the difference between the change in position of the object captured by the camera and change the position of the actual object.

Keyword: Image processing, Stereo Camera Calibration, Non-contact measurement, Object tracking, Measurement.

KATA PENGANTAR

Puji syukur saya panjatkan kehadiran Allah SWT yang telah melimpahkan rahmat dan karuniaNya sehingga saya dapat menyelesaikan Tugas Akhir ini dengan baik. Tidak lupa shalawat serta salam senantiasa tercurahkan kepada Nabi Muhammad Saw. Tugas Akhir ini berjudul:

KALIBRASI STEREO KAMERA DAN PENENTUAN KOORDINAT 3 DIMENSI UNTUK TARGET TUNGGAL PADA SISTEM PELONTAR PELURU AUTOTRACKING.

Tugas Akhir ini ditulis sebagai salah satu syarat kelulusan bagi mahasiswa S1-Teknik Mesin Institut Teknologi Sepuluh Nopember. Dalam kesempatan ini saya menyadari bahwa keberhasilan penyelesaian Tugas Akhir ini tidak lepas dari bantuan dan kerjasama berbagai pihak. Oleh karena itu, saya mengucapkan terimakasih kepada:

1. **Ayahanda tercinta, Almarhum Husnan Lutfi**, atas segala cinta, kasih sayang, dukungan, pengorbanan, dan perjuangan yang telah kau berikan selama hidupmu. Ketiadaanmu menjadi hal terberat dalam hidup saya, namun saya yakin cinta dan kasih sayangmu akan selalu hadir disetiap langkahku. *There's nothing that I want beside you were here*, semua yang tercapai saat ini saya persembahkan untukmu, Ayah.
2. **Ibu tercinta, Choirun Nisa**, atas segala cinta, hiburan, dukungan, candaan dan tetesan air mata di setiap doa yang dipanjatkan untukku senantiasa menemani, mengiringi dan menguatkan saya dalam menjalani setiap masa perkuliahan.
3. **Saudara – saudaraku tersayang, Iqbal Lutfi dan Bilqis Lutfi**, atas segala senyuman, candaan, dan terutama sindiran dan godaan kalian tentang keterlambatan kelulusan saya yang terkadang membangkitkan semangat dan inspirasi saya untuk segera lulus.
4. **Arif Wahyudi, ST, MT, PhD.**, sebagai dosen pembimbing yang telah banyak memberikan pengarahan, bimbingan, dan

waktu di tengah kesibukannya sehingga Tugas Akhir ini selesai.

5. **Dinny Harnany, ST., Msc., Ari Kurniawan Saputra, ST, MT dan Ir. Sampurno, MT.,** selaku dosen penguji yang telah meluangkan waktu untuk menguji dan memberikan arahan untuk kesempurnaan Tugas Akhir ini.
6. **Ir. Bobby Oedy Pramudyo, M.Sc., Ph.D.,** yang telah menjadi dosen wali selama 9 semester, membimbing, memberi arahan dan memberi segenap perhatian layaknya orang tua bagi saya dalam mengarungi perkuliahan di Teknik Mesin ITS.
7. Seluruh Dosen Teknik Mesin ITS, atas segala ilmu dan pengalaman yang diberikan kepada saya.
8. Indira Riska, Dera Fadlih dan Chontry Novita yang selalu hadir dan menghibur disaat stress dan penat selama saya mengerjakan Tugas Akhir ini. Terutama untuk Chontry yang sudah sabar untuk menjadi rekan TA saya meskipun saya yang menyebabkan sempitnya waktu pengerjaan Tugas Akhirnya.
9. TTA (Aishi, Uma, Riska, Salma) yang meskipun saat ini sudah memiliki kesibukan masing – masing, namun sempat mengisi hari – hari saya selama diperkuliahan dulu. Untuk Uma pasti bisa segera menyusul. Doa dan kehadiranku akan selalu menyertai disaat kau membutuhkan.
10. Seluruh teman-teman di Laboratorium Perancangan dan Pengembangan Produk dan warga penghuni Laboratorium Otomasi sementara yang selalu menjadi pengisi keheningan.
11. Mas Tonny, Mas Gagas, Tegar, Mas Andra dan Heri Lutfianto yang tanpa lelah membantu saya dan rekan saya mempelajari hal baru dalam Tugas Akhir ini.
12. Segenap teman-teman M54, angkatan saya, yang telah banyak membantu saya mengarungi bahtera perkuliahan, menjadikan saya mengerti arti tentang persahabatan sejati tanpa mengenal, ras, suku, budaya, adat istiadat, agama. Terima kasih, saya bangga terlahir satu angkatan bersama M54.
13. Pihak-pihak lain yang turut membantu penulis dalam penyelesaian tugas akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih terdapat kekurangan yang dapat digunakan sebagai pertimbangan untuk penelitian selanjutnya. Penulis mengharapkan kritik dan saran untuk perbaikan di masa mendatang. Semoga Tugas Akhir ini bermanfaat bagi kita semua. Aamiin.

Surabaya, 25 Januari 2016

Penulis

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

HALAMAN JUDUL	i
LEMBAR PENGESAHAN	ii
ABSTRAK	iii
ABSTRACT	v
KATA PENGANTAR	vii
DAFTAR ISI	xi
DAFTAR GAMBAR	xiii
BAB I	1
PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	3
1.3 Batasan Masalah.....	3
1.4 Tujuan Penelitian.....	4
1.5 Manfaat Penelitian.....	4
BAB II	5
TINJAUAN PUSTAKA	5
2.1 Tinjauan Pustaka.....	5
2.2 Dasar Teori.....	6
2.2.1. Stereo Vision.....	6
2.2.2. Computer Vision.....	8
2.2.3. Citra Digital.....	8
2.2.4. Kedalaman Citra.....	8
2.2.5. OpenCV.....	14
2.2.6. Koordinat 3 Dimensi.....	15
BAB III	21
METODE PENELITIAN	21
3.1 Flowchart Metodologi Penelitian.....	21
3.2 Tahap Persiapan.....	22
3.3 Tahap Perancangan Program.....	23
3.3.1. Cara Kerja.....	24
3.3.2. Proses dan Pengerjaan.....	25
3.3.3. Flowchart Perancangan Program Kalibrasi Stereo Kamera.....	26
3.3.4. Flowchart Perancangan Program Pendeteksi	

dan Penentu Koordinat	28
3.4 Tahap Implementasi	29
BAB IV	31
PERANCANGAN DAN PEMBUATAN PROGRAM	31
4.1 Implementasi Sistem	31
4.2 Konstruksi Program.....	33
4.2.1. Program Pendeteksian Objek.....	33
4.2.2. Pencarian Variabel Range Warna <i>Threshold</i>	41
4.2.3. Pencarian Variabel Pendeteksian Objek.....	43
4.2.4. Program Penentuan Koordinat x, y, z.....	46
BAB V	31
PENGUJIAN PENDETEKSIAN OBJEK.....	31
5.1 Pendeteksian Objek	48
5.2 Proses Kalibrasi	50
5.3 Pendeteksian Jarak (Koordinat z)	53
5.4 Pendeteksian Jarak Vertikal dan horizontal (Koordinat x dan y)	56
BAB VI	67
KESIMPULAN DAN SARAN.....	67
6.1 Kesimpulan.....	67
6.2 Saran.....	68
DAFTAR PUSTAKA	69
LAMPIRAN	71

DAFTAR GAMBAR

Gambar 1.1 Pemetaan dan pembagian tugas dari Alat Pelontar Peluru	4
Gambar 2.1 Titik Scene dari Kamera Stereo	7
Gambar 2.2 Representasi citra biner.....	10
Gambar 2.3 Citra <i>Grayscale</i>	10
Gambar 2.4 Warna RGB	11
Gambar 2.5 Proses <i>threshold</i>	13
Gambar 2.6 <i>Platform</i> dan OS yang digunakan.....	15
Gambar 2.7 Sistem Koordinat Kartesian 3 Dimensi	16
Gambar 2.8 Hubungan geometrik untuk pengukuran jarak antara optical axis 2 kamera dengan objek	17
Gambar 2.9 Ilustrasi Model Stereo Kamera	18
Gambar 3.1 Diagram alir metodologi peneletian	22
Gambar 3.2 Alat Pelontar Peluru beserta bagian - bagiannya	23
Gambar 3.3 Citra asli dengan citra <i>thresholding</i>	25
Gambar 3.4 Diagram alir program Kalibrasi Stereo Kamera	26
Gambar 3.5 Diagram alir program Pendeteksi dan Pelacak Objek	28
Gambar 4.1 Diagram alir pembuatan program.....	32
Gambar 4.2 Citra RGB sebagai citra masukan frame kiri dan kanan	34
Gambar 4.3 Diagram alir proses pengolahan citra dari RGB dan HSV	35
Gambar 4.4 Diagram alir proses <i>threshold</i>	36
Gambar 4.5 Citra <i>threshold</i>	38
Gambar 4.6 Diagram alir proses <i>erode</i> dan <i>dilate</i>	38
Gambar 4.7 (a) Citra <i>threshold</i> sebelum dilakukan <i>erode dilate</i>	39
Gambar 4.7 (b) Citra <i>threshold</i> hasil <i>erode</i> 1 <i>pixel</i> tetangga	39
Gambar 4.8 Citra <i>threshold</i> hasil <i>dilate</i> 2 tetangga	40

Gambar 4.9 <i>Trackbar</i> yang digunakan untuk mencari Perubahan Citra HSV ke Citra <i>Threshold</i>	41
Gambar 4.10 Citra <i>Threshold</i> dari <i>Trackbar</i>	42
Gambar 4.11 Informasi Hasil Pencarian <i>Range</i> dengan <i>trackbar</i>	42
Gambar 4.12 Diagram alir pencarian koordinat tepi objek	44
Gambar 4.13 Diagram alir pembuatan rectangle.....	45
Gambar 4.14 Diagram alir penentuan nilai x, y, z objek terhadap kamera	46
Gambar 5.1 Hasil pendeteksian objek	48
Gambar 5.2 Grafik hasil pendeteksian objek.....	50
Gambar 5.3 Diagram alir proses kalibrasi	51
Gambar 5.4 Papan catur yang digunakan untuk proses kalibrasi	51
Gambar 5.5 Proses kalibrasi	52
Gambar 5.6 Skema pengambilan data z dan x (tampak atas)	53
Gambar 5.7 Proses pengambilan data	53
Gambar 5.8 Skema pengambilan data z dan y (tampak samping)	54
Gambar 5.9 Grafik antara nilai error z dan nilai z terdeteksi kamera.....	55
Gambar 5.10 Grafik antara nilai error x dan nilai x terdeteksi kamera jarak $z > 290$	57
Gambar 5.11 Grafik antara nilai error x dan nilai x terdeteksi kamera pada jarak $260 < z \leq 290$	58
Gambar 5.12 Grafik antara nilai error x dan nilai x terdeteksi kamera pada jarak $220 < z \leq 260$	58
Gambar 5.13 Grafik antara nilai error x dan nilai x terdeteksi kamera pada jarak $180 < z \leq 220$	59
Gambar 5.14 Grafik antara nilai error x dan nilai x terdeteksi kamera pada jarak $z \leq 180$	59
Gambar 5.15 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $z > 290$	60

Gambar 5.16 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $260 < z \leq 290$	61
Gambar 5.17 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $220 < z \leq 260$	61
Gambar 5.18 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $180 < z \leq 220$	62
Gambar 5.19 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $140 < z \leq 180$	62
Gambar 5.20 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $z \leq 120$	63
Gambar 5.21 Grafik perbandingan nilai x sebenarnya dan nilai x terdeteksi.....	64
Gambar 5.22 Grafik perbandingan nilai y sebenarnya dan nilai y terdeteksi.....	65
Gambar 5.23 Grafik perbandingan nilai z sebenarnya dan nilai z terdeteksi	65

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi saat ini sudah semakin berkembang seiring dengan pesatnya perkembangan jaman. Teknologi pengukuran juga ikut berkembang sehingga pengukuran saat ini menjadi lebih canggih. Pengukuran bertujuan untuk menentukan besaran, dimensi atau kapasitas sesuatu dalam batasan suatu unit standar, biasanya dengan menggunakan instrumen atau proses. Pengukuran juga sebagai suatu hasil, seperti besaran yang mengekspresikan jarak atau nilai yang diperoleh melalui pengukuran. Aplikasi dari pengukuran ini sangat luas, salah satunya bisa digunakan di bidang militer atau persenjataan. Untuk meminimalkan jumlah korban manusia, sistem persenjataan saat ini mulai memanfaatkan mesin yang diharapkan lebih cepat dan lebih akurat sebagai ganti militer manusia. Mesin tersebut bisa dikembangkan dengan memanfaatkan teknologi non contact measurement dan teknologi pendektesian objek yang dapat mendeteksi posisi suatu objek.

Salah satu perkembangan teknologi pengukuran saat ini yaitu pengukuran jarak dengan menggunakan gelombang ultrasonic. Prinsip kerjanya yaitu dengan cara mengirimkan gelombang suara ultrasonik ke objek yang akan diukur dan pantulan gelombang tersebut diterima kembali. Dengan mengetahui kecepatan rambat suara di udara maka dapat diketahui jarak benda dengan cara mengukur waktu dari saat gelombang suara itu dikirim hingga diterima [1]. Jika dibandingkan dengan alat ukur konvensional, alat ukur menggunakan gelombang ultrasonik dapat digunakan tanpa perlu mendekat dan menyentuh objek.

Selain itu, sensor kamera juga bisa dijadikan salah satu cara yang dapat digunakan sebagai alat ukur tanpa perlu kontak langsung dengan objek sekaligus bisa digunakan untuk mendeteksi suatu objek. Deteksi dengan pengolahan citra atau yang biasa disebut *image processing* dapat dilakukan dengan input gambar

maupun video. Pengolahan citra adalah teknik mengolah citra yang mentransformasikan citra masukan menjadi citra lain agar citra keluaran memiliki kualitas yang lebih baik dibandingkan kualitas citra masukan. Pengolahan citra sangat bermanfaat, diantaranya adalah untuk meningkatkan kualitas citra, menghilangkan cacat pada citra, mengidentifikasi objek, dan melakukan penggabungan dengan bagian citra yang lain.

Pada saat ini sudah terdapat alat pelontar peluru yang dapat menembak objek secara otomatis. Dengan memasukkan nilai koordinat dari objek secara otomatis alat tersebut akan menembak kearah yang diinginkan [2]. Namun alat ini memiliki kelemahan yaitu pengguna harus mengetahui koordinat objek kemudian memasukkan nilai koordinat secara manual lewat komputer. Dengan memanfaatkan sensor kamera, suatu aplikasi diusulkan sehingga memiliki kemampuan mengenali objek, persepsi terhadap suatu benda, melakukan pelacakan objek secara *real-time* serta bisa menentukan jarak dari objek tersebut terhadap kamera yang nantinya dapat diaplikasikan pada Alat Pelontar Peluru. Sehingga jika terdapat objek yang tertangkap oleh kamera, aplikasi ini bisa menentukan koordinat 3 dimensi dari objek.



Gambar 1.1 Pemetaan dan pembagian tugas dari Alat Pelontar Peluru *Autotracking*

Seperti yang dapat dilihat pada gambar 1.1, alat pelontar peluru *autotracking* ini terdiri dari berbagai pokok pembahasan. Mulai dari rancang bangun alat pelontar peluru, rancang bangun sistem kontrol dari alat pelontar peluru tersebut, penentuan sistem pengendali PID pada alat pelontar peluru, penambahan sensor kamera yang dapat menentukan koordinat 3 dimensi dari objek, dan rancang bangun sistem alat pelontar peluru berbasis *machine vision*. Keseluruhan pembahasan tersebut yang akan digabungkan menjadi satu untuk menghasilkan suatu alat pelontar peluru *autotracking* yang mampu mendeteksi dan menembak objek secara otomatis dengan sensor kamera.

1.2 Rumusan Masalah

Dalam penyusunan proposal tugas akhir ini, rumusan masalah yang akan dibahas antara lain:

1. Bagaimana mengkalibrasi dua *image* hasil *capture* objek tertentu dari dua kamera yang bersebelahan.
2. Bagaimana menentukan koordinat 3 dimensi dari objek terhadap kamera.

1.3 Batasan Masalah

Untuk menyederhanakan Tugas Akhir ini, masalah dibatasi sebagai berikut:

1. Perangkat lunak yang digunakan adalah perangkat lunak lengkap (*suite*) microsoft visual studio 2010.
2. Menggunakan teknologi *computer vision* dengan openCV 2.1 sebagai *library* algoritma bahasa C++.
3. Kamera yang digunakan posisinya tidak bergerak dengan tipe *CMOS* (*Complementary Metal-Oxide Semiconductor*).
4. Menggunakan 2 kamera (*stereo vision*).
5. Hanya bisa mendetekteksi objek tunggal.
6. Objek yang dideteksi tidak bergerak.
7. Kemampuan pendeteksian sangat dipengaruhi pencahayaan dan spesifikasi kamera.

8. Objek yang dipakai adalah bola berdiameter 8 cm dan berwarna kuning.
9. Jarak minimal pendeteksian yaitu 80 cm dan jarak maksimal sejauh 5 meter.
10. Menggunakan sistem pencahayaan *indoor* pada waktu siang hari.
11. Terdapat *blind area* diantara 2 kamera.

1.4 Tujuan Penelitian

Tujuan dari proposal tugas akhir ini adalah sebagai berikut:

1. Mengetahui cara mengkalibrasi dua *image* hasil *capture* objek tertentu dari dua kamera yang bersebelahan.
2. Dapat menentukan koordinat 3 dimensi dari objek terhadap kamera.

1.5 Manfaat Penelitian

Manfaat dari proposal tugas akhir ini adalah sebagai berikut:

1. Menambah *database* tentang *image processing* sebagai penginderaan jarak jauh dan mendeteksi objek.
2. Sebagai bahan referensi bagi penelitian sejenisnya dalam rangka pengembangan pengetahuan tentang *image processing*.
3. Sebagai masukan dalam pengembangan otomatisasi mesin-mesin industri manufaktur dan militer.
4. Dapat digunakan sebagai alat ukur jarak jauh.

BAB II TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Dengan kemajuan teknologi pendeteksian dan pelacakan objek yang semakin pesat, maka kehidupan manusia dapat dipermudah, dan saat ini banyak aplikasi yang menerapkannya dalam berbagai bidang. Mulai dari pendeteksian objek tunggal sampai dengan multi objek sudah diusulkan pada *paper – paper* beberapa tahun terakhir. Banyak *paper* dengan berbagai metode yang dapat digunakan untuk mendeteksi suatu objek tunggal dan menentukan koordinat 3 dimensi dari suatu objek.

Terdapat penelitian terdahulu yang membahas tentang pelacakan dan pendeteksian 2D suatu objek. Penelitian ini menggunakan metode multi objek dengan *hierarchical particle filter*. Metode ini memanfaatkan acuan range warna yang ada pada objek di setiap *frame*-nya. Pelacakan membutuhkan 1000 *frame* sampel dengan kecepatan 50 *fps* (*frame per second*) dan ukuran gambar yang dipakai yaitu berukuran 352 x 288 *pixels*. Penelitian ini menggunakan objek berupa bola dan dilakukan pendekatan histogram warna RGB (*red green blue*) dari setiap *frame*-nya. Namun penelitian ini memiliki kekurangan yaitu hanya bisa menentukan koordinat 2D dari objek dikarenakan hanya menggunakan kamera tunggal. Sehingga tidak bisa menentukan jarak antara objek terhadap kamera [3].

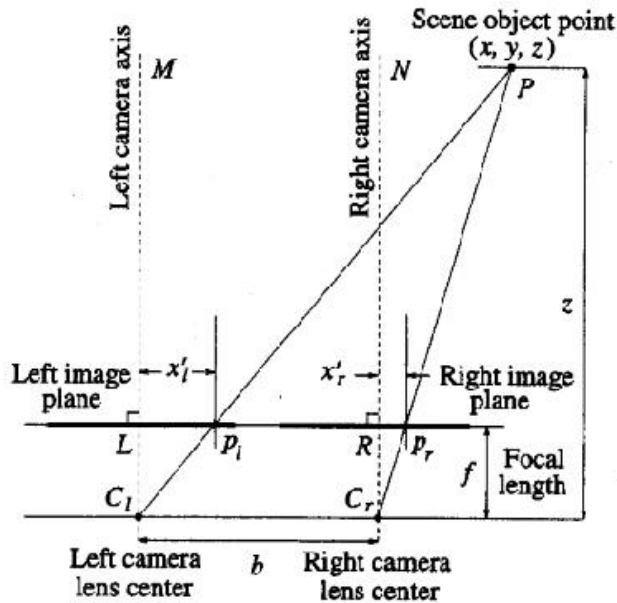
Terdapat juga riset untuk menentukan koordinat objek yang dilakukan dengan memanfaatkan pengolahan citra hasil dari kamera digital. Metode yang dilakukan pada penelitian ini yaitu dengan pendekatan proyeksi perspektif hasil kamera yang dapat dirumuskan secara matematis menggunakan dua titik hilang. 4 titik acuan dibutuhkan untuk menghasilkan dua titik hilang. Garis horizon atau tinggi mata akan dihasilkan dari dua titik hilang, titik hilang yang melalui titik yang dituju dapat diproyeksikan ke garis tanah yang menjadi acuan ukuran yang sebenarnya [4].

Terdapat juga penelitian yang membahas tentang pengukuran jarak suatu objek dengan menggunakan kamera. Penelitian ini menghasilkan suatu program yang mampu mengestimasi jarak suatu objek yang berupa bola terhadap kamera stereo. Penelitian ini memanfaatkan stereo kamera untuk menentukan jarak suatu objek terhadap kamera. Parameter penting yang harus dicari dalam perhitungan algoritma *stereo vision* yaitu parameter *focal length*, parameter ini didapat melalui proses kalibrasi stereo kamera. *Focal length* ini menentukan kuat atau lemahnya suatu kamera untuk memfokuskan atau menyimpangkan cahaya. Penelitian ini menggunakan perangkat lunak *Visual Studio* yang ditulis dalam bahasa C dari library *OpenCV*. Kelemahan dari penelitian ini program hanya mampu mendeteksi dan mengestimasi jarak suatu objek dari jarak tertentu, yaitu jarak minimal 50 cm dan jarak maksimal 300 cm. untuk jarak diatas 300 cm objek akan sulit terdeteksi atau bahkan tidak terdeteksi sama sekali [5].

2.2 Dasar Teori

2.2.1 Stereo vision

Stereo vision adalah suatu teknik untuk mendapatkan sebuah impresi kedalaman dari alat penglihatan ke benda yang dituju dari alat tersebut. Teknik ini mampu memungkinkan seorang pengguna untuk menentukan jarak dari alat penglihatan terhadap suatu objek yang ditentukan. Banyak sekali manfaat yang dihasilkan dari stereo vision, dikarenakan teknik ini mampu mengestimasi jarak suatu objek, salah satunya dengan mengimplementasikan hasil pengukuran jarak ini pada pengembangan sensor dan robotika. Kamera secara geometri memiliki fungsi untuk memetakan objek pada ruang 3 dimensi menjadi bidang 2 dimensi dalam bentuk gambar atau citra. Pemetaan koordinat ruang 3D menjadi 2D ini tidak cukup sederhana tetapi melibatkan banyak variabel yang harus dipertimbangkan.



Gambar 2.1 Titik Scene dari kamera stereo

Sumber : R. Jain, 1995, "Machine Vision"

Stereo vision dengan 2 kamera atau lebih untuk dapat mengukur jarak objek juga perlu dikalibrasi. Banyak variabel penting yang diperlukan untuk dapat menentukan jarak objek. Seperti yang sudah digambarkan pada gambar 2.1 ditunjukkan berbagai variabel – variabel penting yang diperlukan. Salah satunya yaitu *focal length* (f). *Focal length* yaitu ukuran jarak antara elemen lensa dengan permukaan film (sensor digital) pada kamera. Variabel ini didapat dari hasil kalibrasi stereo kamera yang berupa matrik yang berisi nilai *focal lengt* dan pusat lensa tiap kamera (C_l dan C_r). Selain itu terdapat juga variabel penting lainnya yang digunakan untuk menentukan jarak objek terhadap kamera seperti b yang merupakan jarak kedua pusat proyeksi kamera, lalu ada juga x'_l x'_r yang merupakan jarak antara

pusat proyeksi kamera dengan garis imajinasi dan juga besar sudut yang dibentuk oleh arah pandang kedua kamera [6].

2.2.2 *Computer Vision*

Computer Vision merupakan sebuah disiplin ilmu yang mempelajari bagaimana menginterpretasikan dan merekonstruksi sebuah tampilan tiga dimensi dari bentuk dua dimensi. *Computer Vision* ini bertujuan untuk memodelkan dan meniru pengelihatan manusia dengan memanfaatkan *software* dan *hardware* pada komputer. *Computer Vision* menggabungkan berbagai bidang ilmu pengetahuan untuk bisa memahami dan menyimulasikan pengoprasian sistem pengelihatan manusia[7].

Computer vision memiliki banyak pendukung yang dibutuhkan untuk bisa berfungsi secara baik, ini dilakukan agar *computer vision* mampu menangkap informasi secara maksimal. Fungsi – fungsi tersebut diantaranya yaitu [8]:

1. *Image acquisition*
2. *Image processing*
3. *Image analysis*
4. *Image understanding*

2.2.3 *Citra Digital*

Citra merupakan representasi dua dimensi dari bentuk fisik nyata 3 dimensi, yang mana perwujudannya bisa bermacam – macam. Banyak bentuk dari citra, mulai dari gambar hitam putih pada sebuah foto yang tidak bergerak sampai pada gambar berwarna yang bergerak pada sebuah televisi. Citra dihasilkan dari gambar analog dua dimensi dan kontinu menjadi gambar diskrit, melalui proses sampling gambar analog dibagi menjadi M baris dan N kolom sehingga menjadi gambar diskrit [9]. Sedangkan citra digital merupakan citra dua dimensi yang dapat ditampilkan pada layar komputer dalam bentuk *pixel* picture elements.

Oleh karena citra merupakan matrik dua dimensi dari fungsi intensitas cahaya, maka dua variabel yang menunjukkan posisi pada bidang dengan sebuah fungsi intensitas cahaya dipakai dan dapat dituliskan dengan fungsi $f(x,y)$ yang mana nilai f merupakan nilai amplitudo pada koordinat spasial (x,y) . Citra digital juga bisa digambarkan sebagai matriks $M \times N$ yang baris dan kolomnya menunjukkan titik – titik *pixel*, dimana bentuk matriks dari citra digital dapat digambarkan dalam bentuk matematis seperti persamaan (1):

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \dots\dots\dots(1)$$

2.2.4 Kedalaman Citra

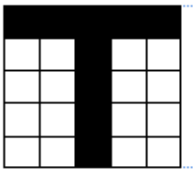
Nilai yang dapat digunakan untuk menguantisasi citra bergantung kepada kedalaman *pixel*. *Pixel* merupakan unsur gambar atau representasi sebuah titik terkecil dalam sebuah gambar grafis yang dihitung per inci [10]. Semakin tinggi jumlah *pixel* pada suatu gambar maka akan semakin tinggi resolusinya. Selain *pixel*, jumlah bit juga ikut mempengaruhi kedalaman citra. Berapa banyaknya bit yang digunakan untuk merepresentasikan intensitas warna *pixel*. Citra digital yang mempunyai kedalaman *pixel* n bit disebut sebagai citra n -bit.

Sedangkan berdasarkan warna – warna penyusunnya, citra digital dapat dibagi menjadi tiga macam, yaitu [11]:

1. Citra Biner

Citra biner yaitu citra yang hanya memiliki dua warna, yaitu hitam dan putih. Sehingga setiap *pixel* pada citra biner cukup direpresentasikan dengan 1 bit. Contoh citra biner adalah pada gambar 2.2 :

1	1	1	1	1
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0



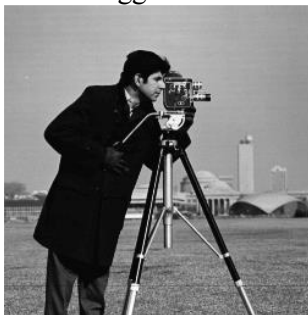
Gambar 2.2 Representasi citra biner

Sumber : <https://ginan88.files.wordpress.com/2011/12/2.png>

Seperti yang dapat dilihat pada gambar 2.2, *Pixel* – *pixel* dari objek bernilai 1 dan *pixel* – *pixel* latar belakang bernilai 0. Sehingga pada waktu menampilkan gambar, *pixel* yang bernilai 0 menjadi warna putih dan *pixel* yang bernilai 1 menjadi warna hitam.

2. Citra *Grayscale*

Citra *grayscale* merupakan citra yang nilai *pixel*nya menunjukkan derajat keabuan atau intensitas warna putih. Nilai intensitas paling rendah akan menghasilkan warna hitam dan nilai intensitas paling tinggi akan menghasilkan warna putih. Kebanyakan citra *grayscale* memiliki kedalaman *pixel* 8 bit (256 derajat keabuan), tetapi terdapat juga citra *grayscale* yang kedalaman *pixel*nya bukan 8 bit, misalnya 16 bit yang biasa digunakan pada citra yang memerlukan ketelitian tinggi.

Gambar 2.3 Citra *Grayscale*

Sumber : <http://dani-komputer.blogspot.co.id/2013/05/program-mengubah-citra-grayscale-ke.html>

Contohnya adalah pada gambar 2.3, pada gambar tersebut tiap objek hanya dibedakan melalui derajat keabuan, ada bagian yang memiliki derajat keabuan rendah hingga hampir mendekati hitam dan juga bagian yang memiliki nilai derajat keabuan tinggi sehingga warnanya mendekati warna putih.

Untuk merubah suatu citra dengan warna penuh (RGB) menjadi suatu citra *grayscale* (gambar keabuan), terdapat metode yang umum digunakan, yaitu:

$$\frac{R+G+B}{3} \dots\dots\dots(2)$$

dimana : R : Unsur warna merah

G : Unsur warna hijau

B : Unsur warna biru

Dengan mengaplikasikan metode tersebut pada tiap *pixel* yang ada pada suatu citra maka akan didapatkan citra *grayscale*.

3. Citra Warna

Citra berwarna merupakan citra yang direpresentasikan dalam beberapa *channel* yang menggambarkan komponen – komponen warna penyusunnya. Banyaknya *channel* yang dipakai bergantung pada model warna yang digunakan pada citra tersebut. Sistem grafik yang digunakan untuk menggambarkan citra warna memiliki satu set nilai yang tersusun dan menyatakan berbagai tingkat warna. Setiap *pixel* pada citra warna diwakili oleh warna yang merupakan kombinasi dari tiga warna dasar merah hijau biru atau *Red Green Blue* (RGB).



Gambar 2.4 Warna RGB

Sumber : <https://pixabay.com/en/intersection-mix-colors-rgb-red-154782/>

Citra warna (8 bit) merupakan citra yang tiap *pixel*nya memiliki jumlah warna maksimum yang dapat digunakan sebanyak 256 warna. Setiap *pixel* pada citra warna diwakili oleh warna yang merupakan kombinasi dari tiga warna dasar yaitu merah, hijau, dan biru yang biasa disebut citra RGB (Red, Green, Blue).

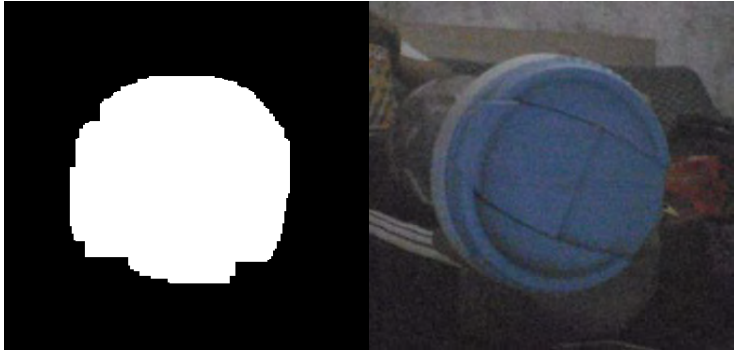
Citra warna (16 bit) atau citra *highcolor* merupakan citra yang tiap *pixel*nya diwakili dengan 65.536 warna (2 byte memory). Dalam formasi bitnya, nilai merah dan biru mengambil tempat di 5 bit di kanan dan kiri. Komponen hijau memiliki 5 bit ditambah 1 bit ekstra.

Citra warna (24 bit) merupakan citra yang tiap *pixel*nya diwakili dengan 16.777.216 variasi warna. Variasi tersebut sangat mampu untuk memvisualisasikan seluruh warna yang dapat dilihat oleh penglihatan manusia. Setiap poin informasi *pixel* (RGB) disimpan ke dalam 1 byte data dengan 8 bit pertama menyimpan nilai biru, kemudian diikuti dengan nilai hijau pada 8 bit kedua dan pada 8 bit terakhir merupakan warna merah.

Berdasarkan sumber lain, selain 3 macam citra yang sudah disebutkan diatas, terdapat beberapa macam citra digital lainnya, diantaranya :

Citra threshold

Citra *threshold* dilakukan bertujuan untuk mempertegas citra dengan cara mengubah citra hasil yang memiliki derajat keabuan 255 (8 bit), menjadi citra biner atau citra yang memiliki hanya dua warna yaitu hitam dan putih. Hal yang harus diperhatikan pada saat melakukan proses *threshold* yaitu saat memilih sebuah nilai *threshold* (T) dimana *pixel* yang bernilai dibawah nilai *threshold* akan diubah menjadi hitam dan *pixel* yang bernilai diatas nilai *threshold* akan diubah menjadi putih. Seperti yang dapat dilihat pada gambar 2.5



Gambar 2.5 Proses *threshold*

Sumber : Dokumentasi pribadi

HSV

Pada pengolahan suatu warna citra terdapat bermacam – macam model warna. Model dengan menggunakan warna dasar merah, hijau, biru atau yang biasa disebut RGB (*red green blue*) merupakan model yang paling sering digunakan, salah satunya seperti yang digunakan pada layar monitor dan televisi. Pada model ini 3 buah komponen warna tersebut untuk merepresentasikan suatu citra [12]. Selain model RGB terdapat juga model HSV dimana model ini terdapat 3 komponen yaitu, *hue*, *saturation*, dan *value* [13].

1. *Hue* merupakan suatu ukuran panjang gelombang yang terdapat pada warna dominan yang diterima oleh penglihatan. Hue menyatakan warna yang sebenarnya, seperti merah, violet, kuning dan digunakan untuk menentukan kemerahan (*redness*), kehijauan (*greenness*) dan sebagainya.
2. *Saturation* atau *chroma* adalah kemurnian atau kekuatan warna, ukuran banyaknya cahaya putih yang bercampur pada *hue*.

3. *Value* adalah nilai kecerahan dari warna. Nilainya mulai dari 0 – 100 %. Apabila nilainya 0 maka warna yang dihasilkan menjadi hitam, semakin besar nilai maka semakin cerah dan muncul variasi – variasi baru dari warna tersebut.

Untuk mengubah citra HSV menjadi citra RGB bisa digunakan persamaan seperti berikut ini:

$$\begin{aligned}
 V &= \max(r, g, b) \\
 S &= \begin{cases} 0, & v = 0 \\ 1 - \frac{\min(r, g, b)}{v}, & v > 0 \end{cases} \\
 H &= \begin{cases} 0, & \text{jika } S = 0 \\ \frac{60 * (g - b)}{S * v}, & \text{jika } v = r \\ 60 * \left[2 + \frac{b - r}{S * v} \right], & \text{jika } v = g \\ 60 * \left[4 + \frac{r - g}{S * v} \right], & \text{jika } v = b \end{cases} \dots\dots\dots (3)
 \end{aligned}$$

Jika Saturation $S=0$, maka hue tidak terdefinisi atau dengan kata lain tidak memiliki hue berarti *monochrome*. Hue (H) lalu dikonversi menjadi derajat/degrees dengan cara mengalikan dengan 60 sehingga menghasilkan HSV dengan S dan V antara 0 dan 1 dan H antara 0 – 360.

2.2.5 OpenCV

OpenCV (*Open Source Computer Vision*) merupakan sebuah *Open Source BSD-licensed library* yang berisi lebih dari 500 algoritma yang telah dioptimalisasi dengan baik untuk pengolahan gambar dan video analisis [14]. Aplikasi ini bisa didapatkan didapatkan dari <http://SourceForge.net/projects/opencvlibrary>. *Library* ini ditulis dengan bahasa C dan C++ dan dapat digunakan dengan *operating system* Windows, Linux, dan Mac OS X. OpenCV dirancang untuk efisiensi komputasional.

Tujuan dari aplikasi ini yaitu agar komputer memiliki kemampuan yang mirip dengan cara pengolahan

visual pada manusia. *Library* ini dibuat untuk bahasa C/C++ sebagai optimasi aplikasi *real-time*. OpenCV memiliki API yang merupakan singkatan dari *Application Programming Interface* yang digunakan untuk pengolahan tingkat tinggi maupun tingkat rendah. Pada OpenCV juga terdapat fungsi – fungsi siap pakai untuk mengambil, menyimpan, serta mengakuisisi gambar dan video.



Gambar 2.6 Platform dan OS yang digunakan

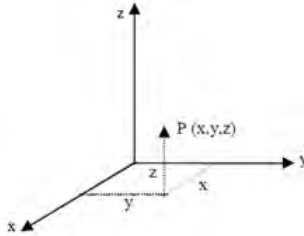
Sumber : <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS4818nKZE-nPln-Wl55I9Y2tQPtOtTlt6HEncQnVvMovloHDJ6Qw>

OpenCV dapat dijalankan diberbagai platform. Platform yang biasa digunakan yaitu dengan menggunakan Eclipse (minimum versi OpenCV 2.0), Microsoft Visual studio 2008 hingga Microsoft Visual Studio 2013 (minimum versi OpenCV 2.0), Android (minimum versi OpenCV 2.4.2), Java (minimum versi OpenCV 2.4.4), iOS (minimum versi OpenCV 2.4.2), Windows (minimum versi OpenCV 2.0), Linux (minimum versi OpenCV 2.0).

2.2.6 Koordinat 3 Dimensi

Sistem koordinat Kartesius dalam dua dimensi didefinisikan dengan dua sumbu yang saling tegak lurus antar satu dengan yang lain, serta keduanya terletak pada satu bidang (bidang xy), seperti yang dapat dilihat pada gambar 2.7. Sumbu horizontal diberi label x, dan sumbu vertikal diberi label y. Pada sistem koordinat tiga dimensi, ditambahkan sumbu yang lain yang sering diberi label z. Sumbu-sumbu tersebut ortogonal (satu sumbu dengan

sumbu lain bertegak lurus) antar satu dengan yang lain [15].



Gambar 2.7 Sistem Koordinat Kartesian 3 Dimensi

Sumber : <https://zenosoft.files.wordpress.com/2014/05/42.png>

Untuk menentukan koordinat 3 dimensi dari objek yang tertangkap kamera dapat digunakan metode *Principle of Distance Measurement*. Pada metode ini jarak Dz bisa dicari melalui melalui persamaan [17]:

$$Dz = \frac{L}{\tan P_L + \tan P_R} \dots\dots\dots(4)$$

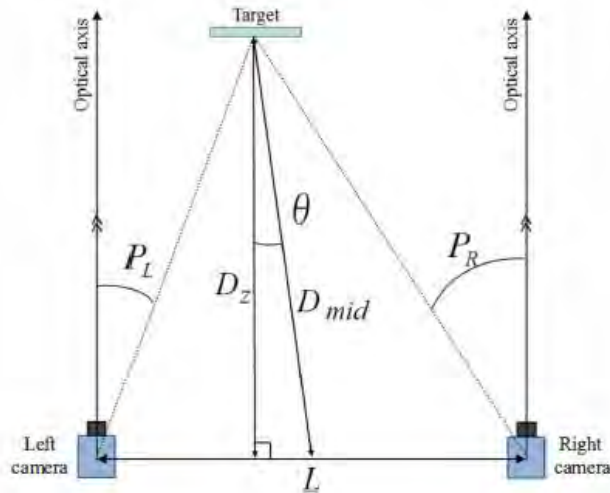
Dimana :

L = Jarak antara 2 kamera

P_L = Sudut antara optical axis kamera kiri dengan garis lurus kamera kiri terhadap objek

P_R = Sudut antara optical axis kamera kanan dengan garis lurus kamera kanan terhadap objek

Untuk lebih jelas dapat dilihat pada gambar 2.8, untuk mencari nilai Dz (jarak objek terhadap kamera).



Gambar 2.8 Hubungan geometrik untuk pengukuran jarak antara *optical axis* 2 kamera dengan objek

Sumber : Tadashi Ogura, Yoshiaki Mizuchi, dkk. (2013)

Besar sudut P_L dan P_R bisa didapat melalui persamaan :

$$P = \tan^{-1} \frac{u_t p}{f} \dots\dots\dots(5)$$

$$T = \tan^{-1} \frac{v_t p}{f} \dots\dots\dots(6)$$

Dimana :

u_t = titik pusat kamera (koordinat x)

v_t = titik pusat kamera (koordinat y)

p = ukuran *pixel*

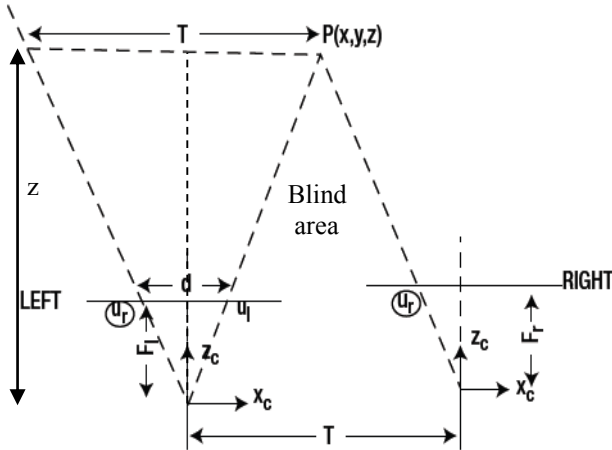
f = *focal length* dari kamera

Sehingga jarak objek terhadap titik tengah kamera bisa dicari melalui persamaan:

$$D_{mid} = \sqrt{\frac{1}{2} D_z^2 + (\tan^2 P_L + \tan^2 P_R + 2) - \frac{L^2}{4}} \dots\dots(7)$$

Selain menggunakan persamaan – persamaan diatas, jarak antara objek dengan kamera juga bisa didapat

dengan memanfaatkan persamaan segitiga sebangun[18]. Seperti yang dapat dilihat pada gambar 2.9.



Gambar 2.9 Ilustrasi model stereo kamera

Sumber : Samarth Brahmabhatt. (2013)

$$\frac{d}{T} = \frac{f}{z} \dots \dots \dots (8)$$

Dimana :

f_l = focal length kamera kiri

f_r = focal length kamera kanan

T = jarak antar dua titik pusat kamera

$$d = u_l - u_r \dots \dots \dots (9)$$

sehingga,

$$z = \frac{f * T}{d} \dots \dots \dots (10)$$

Namun dalam penggunaan dua kamera ini terdapat satu kelemahan yaitu terdapat blind area diantara dua kamera. Dapat dilihat pada gambar 2.9. Semakin dekat jarak anatar dua pusat kamera, blind area pun akan semakin kecil. Serta area tangkap dimana pada area tersebut objek bisa dideteksi jaraknya juga akan semakin luas.

Dari nilai z yang sudah didapat, koordinat x dan y yang merupakan jarak vertikal dan horizontal objek terhadap kamera dapat diketahui dengan menggunakan persamaan yang memanfaatkan segitiga sebangun seperti berikut :

$$\frac{ul}{x} = \frac{f}{z} \dots\dots\dots(11)$$

$$x = \frac{z*ul}{f} \dots\dots\dots(12)$$

$$\frac{vl}{y} = \frac{f}{z} \dots\dots\dots(13)$$

$$y = \frac{z*vl}{f} \dots\dots\dots(14)$$

Dimana :

f_l = *focal length* kamera kiri

f_r = *focal length* kamera kanan

T = jarak antar dua titik pusat kamera

u_l = posisi objek yang tertangkap kamera kiri
pada sumbu x

u_r = posisi objek yang tertangkap kamera kanan
pada sumbu x

z = jarak objek tegak lurus terhadap kamera

v_l = posisi objek yang tertangkap kamera kiri
pada sumbu y

v_r = posisi objek yang tertangkap kamera kanan
pada sumbu y

(Halaman ini sengaja dikosongkan)

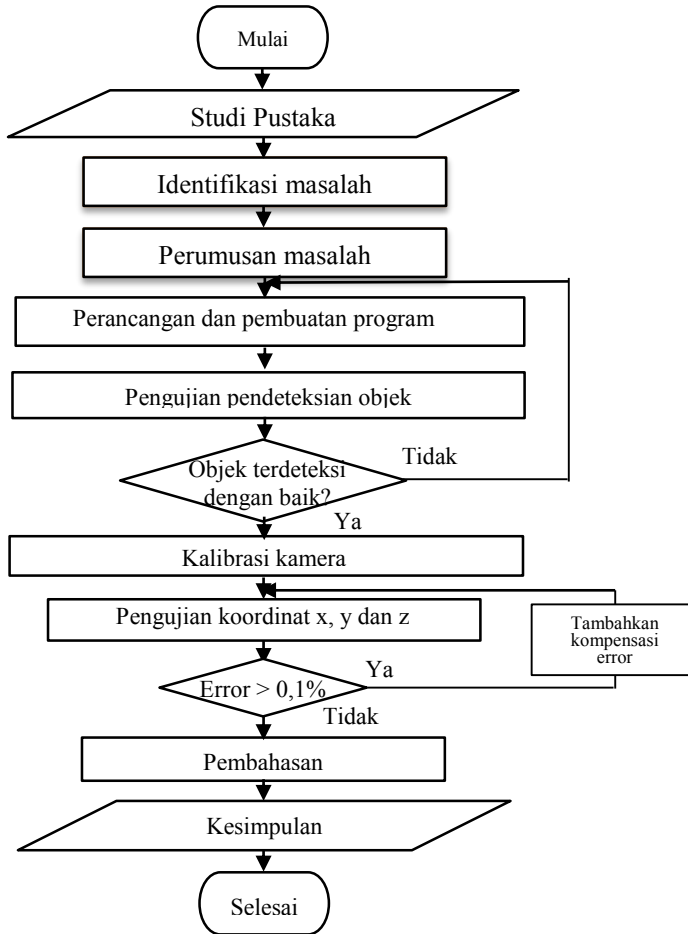
BAB III

METODE PENELITIAN

Pada Bab III ini akan dipaparkan mengenai langkah-langkah sistematis dan terarah yang akan dijadikan sebagai acuan kerangka penelitian yang bertujuan untuk menciptakan aplikasi pendeteksi, pelacak dan penentu koordinat 3 dimensi suatu objek. Selain itu juga dijelaskan parameter proses yang digunakan dalam penelitian ini.

3.1 Flowchart Metodologi Penelitian

Secara garis besar metodologi yang digunakan dalam penelitian ini dapat dibagi menjadi tiga tahapan utama yaitu tahap persiapan, tahap perancangan program, serta tahap analisa yang berisi pembahasan dan penarikan kesimpulan seperti yang terdapat pada gambar diagram alir 3.1. Tahap persiapan terdiri dari studi pustaka yang berujuan untuk mempelajari dasar – dasar dari penelitian, kemudian dillakukan identifikasi masalah dan perumusan masalah. Setelah tahap persiapan selesai, dilanjutkan dengan tahap perancangan. Setelah didapat hasil dari tahap tersebut, dapat dilakukan pembahasan hasil dari pengujian yang sudah dilakukan sehingga dapat diambil kesimpulan dari Tugas Akhir ini.



Gambar 3.1 Diagram alir metodologi penelitian

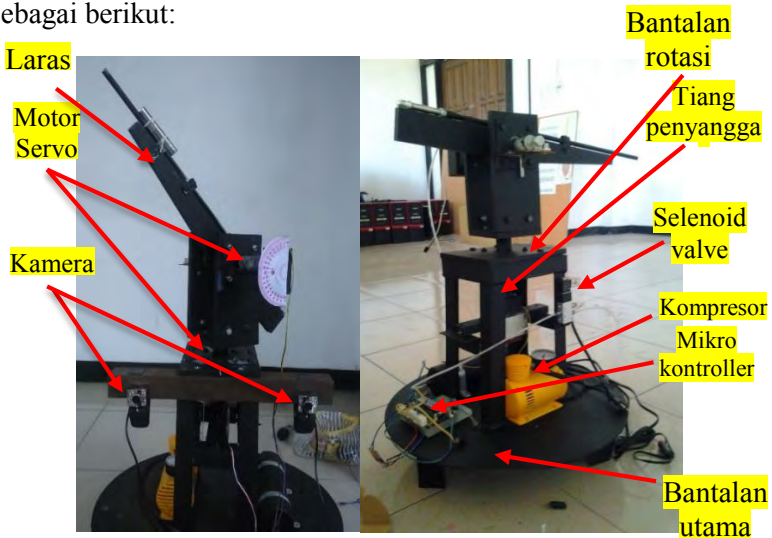
3.2 Tahap Persiapan

Tahap persiapan ini terdiri dari tiga tahapan yaitu studi pustaka, identifikasi masalah dan perumusan masalah. Studi pustaka adalah acuan referensi penulis untuk mendalami permasalahan yang akan diteliti mengenai pembuatan aplikasi program pendeteksi, pelacak dan penentu koordinat 3 dimensi

suatu objek dari buku-buku referensi dan jurnal yang berkaitan dengan permasalahan yang akan dibahas. Kemudian dilakukan identifikasi masalah untuk menyusun sistem pendeteksi, pelacak dan penentu koordinat 3 dimensi suatu objek. Sedangkan perumusan masalah mencakup perancangan program untuk pendeteksi, pelacak dan penentu koordinat 3 dimensi suatu objek.

3.3 Tahap Perancangan Program

Pada tahap ini dilakukan perancangan program kalibrasi stereo kamera, kemudian dilakukan perancangan program untuk mendeteksi, melacak dan menentukan koordinat x , y , dan z objek terhadap dua kamera. Sebelum membahas tentang perancangan program yang akan dilakukan, dapat dilihat skema dari alat pelontar peluru dimana program pendeteksi, pelacak dan penentu koordinat 3 dimensi ini akan diaplikasikan pada gambar 3.2 sebagai berikut:



Gambar 3.2 Alat Pelontar Peluru beserta bagian – bagiannya

Seperti yang terlihat pada gambar 3.2 bahwa Alat Pelontar Peluru tersebut terdiri dari berbagai bagian seperti selongsong

peluru sebagai pengarah, landasan utama, bantalan rotasi, motor dan lain-lain. Bantalan utama berupa plat besi setebal 50 mm yang memiliki fungsi sebagai pondasi tempat dipasangnya tiang penyangga untuk bantalan rotasi dan tempat dipasangnya *solenoid valve*. Bantalan rotasi pada pelontar peluru berhubungan langsung dengan motor penggerak pelontar ke arah sumbu x. Sedangkan fungsi dari motor yaitu untuk menghubungkan antara bantalan rotasi dengan papan penyangga penembak. Motor penggerak dipasang pada sisi kiri pangkal laras supaya laras dapat bergerak naik turun (membentuk sudut elevasi). Secara garis besar pelontar peluru tersebut akan bekerja secara otomatis dengan inputan berupa *image target* yang kemudian diolah dengan menggunakan program yang dihasilkan dari Tugas Akhir ini. Masukan yang dihasilkan oleh Tugas Akhir ini berupa koordinat x, y dan z objek. Kemudian oleh rekan saya, kooordinat posisi dari objek kemudian dikomunikasikan ke mikrokontroller. Mikrokontroller yang mendapatkan informasi kemudian memberi perintah kepada motor servo untuk bergerak, sehingga pelontar peluru mampu bergerak sesuai dengan kebutuhan.

3.3.1 Cara Kerja

Cara kerja yang harus dilakukan yaitu dengan mendeteksi objek yang merupakan bola berwarna kuning dalam bentuk *image*. Kemudian *image* ini dianalisa dan diproses menggunakan program C++ dari *library* openCV 2.1.0 dan *Software* Visual Studio 2010. Setelah itu dilakukan proses kalibrasi kamera untuk mengetahui nilai intrinsik dua kamera (*focal length*, u dan v). Dari hasil kalibrasi nilai z dapat ditentukan dengan menggunakan perbandingan segitiga sebangun. Dari nilai z ini akan didapat koordinat x dan y objek terhadap kamera. Hasil koordinat x, y dan z diserialkan agar bisa diaplikasikan pada Alat Pelontar Peluru.

3.3.2 Proses dan Pengerjaan

Ada beberapa tahapan dalam proses dan pengerjaan tugas pada perancangan program yaitu pendektesian dan pelacakan objek, penghitungan jarak objek terhadap 2 kamera (koordinat z) dan penentuan koordinat x dan y objek terhadap 2 kamera.

1. Kalibrasi stereo kamera

Program untuk mengkalibrasi kamera dibuat untuk mengetahui nilai intrinsik tiap kamera. Data – data yang didapat yaitu *focal length* (f), *imaging 2D coordinate system* (u, v), titik gambar 2D kamera kanan dan kiri, dan matriks kamera kanan dan kiri (opsional).

2. Pendektesian, Pelacakan dan Penentuan koordinat x, y dan z Objek

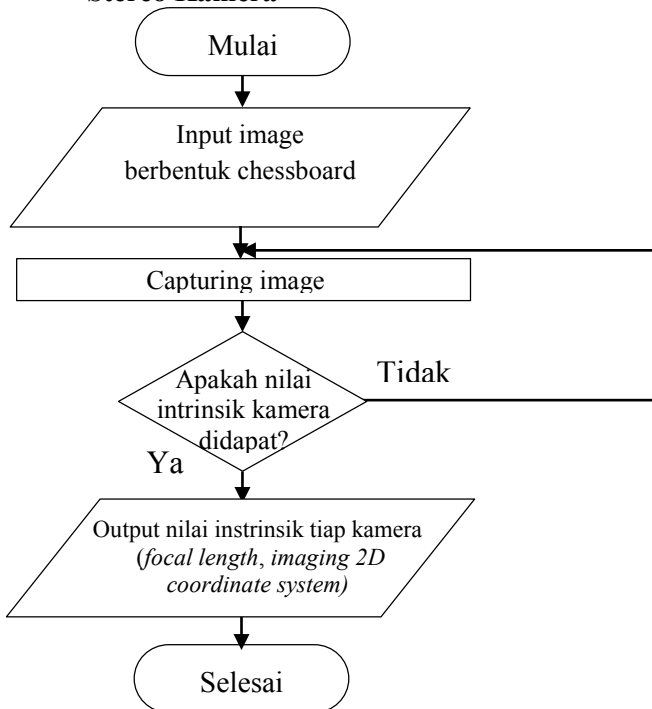
Pendektesian dan Pelacakan Objek dilakukan dengan cara mengatur nilai warna sesuai yang diinginkan pada program agar pada saat program dijalankan bisa mendeteksi dan melacak objek yang memiliki warna tertentu. Langkah – langkah yang perlu dilakukan yaitu mengambil citra yang didalamnya terdapat objek tertentu. Kemudian dilakukan proses untuk mengubah citra yang sebelumnya RGB ke HSV. Citra yang sudah berupa HSV dilakukan *thresholding* untuk menghasilkan citra biner dimana warna kuning menjadi warna hitam dan warna lainnya menjadi warna putih. Seperti yang ditunjukkan pada gambar berikut.



Gambar 3.3 Citra asli dengan citra hasil *threshold*

Penentuan koordinat x , y dan z objek terhadap 2 kamera dilakukan dengan memanfaatkan nilai hasil dari kalibrasi. Kalibrasi cukup dilakukan sekali. Sehingga jika objek digerakkan menjauh maupun mendekat terhadap kamera jarak akan diketahui. Jika koordinat z sudah diketahui, koordinat x dan y bisa ditentukan juga dengan menggunakan persamaan. Nantinya koordinat x , y dan z dari objek terhadap 2 kamera diserialkan agar bisa diproses untuk menjadi data masukan pada Alat Pelontar Peluru.

3.3.3 Flowchart Perancangan Program Kalibrasi Stereo Kamera

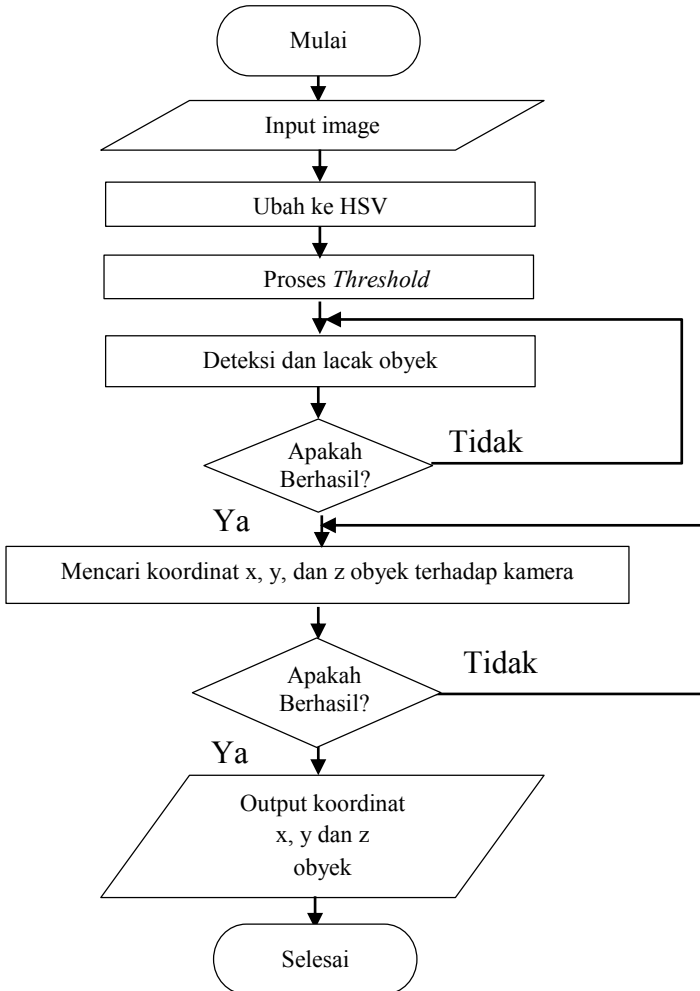


Gambar 3.4 Diagram alir rancangan program Kalibrasi Stereo Kamera

Seperti yang dapat dilihat pada gambar 3.4, proses kalibrasi ini digunakan program kalibrasi yang bertujuan untuk mengetahui nilai instrinsik kamera. Hasil dari kalibrasi ini berupa matrik 3x3 yang berisi nilai *focal length* serta titik pusat tiap kamera.

$$K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \dots\dots\dots(10)$$

3.3.4 Flowchart Perancangan Program Pendeteksi dan Penentu Koordinat



Gambar 3.5 Diagram alir rancangan program Pendeteksi dan Pelacak Obyek

Pada diagram alir 3.5 dapat dilihat bahwa proses yang pertama kali dilakukan yaitu melakukan image processing. Citra inputan yang berupa citra RGB dirubah terlebih dahulu menjadi citra HSV. Kemudian dari hasil citra tersebut dilakukan proses *thresholding* untuk merubah citra menjadi biner, dimana objek diubah menjadi warna hitam dan latar belakang dirubah menjadi warna putih. Dari proses tersebut objek akan terdeteksi. Dari hasil pendeteksian tersebut dapat diperoleh koordinat 3 dimensi dari objek.

3.4 Tahap Implementasi

Tahap Implementasi merupakan tahap implementasi dari *software* yang telah dirancang. *Running* program dengan berbagai variasi posisi objek. Kemudian dari hasil deteksi objek tersebut sehingga didapatkan data berupa koordinat x , y dan z objek dari berbagai posisi. Data ini dijadikan bahan analisa untuk menentukan persamaan hubungan antara koordinat objek terhadap 2 kamera dan koordinat objek terhadap laras Alat Pelontar Peluru.

(Halaman ini sengaja dikosongkan)

BAB IV

PERANCANGAN DAN PEMBUATAN PROGRAM

4.1. Implementasi Sistem

Program pendeteksi objek ini dibuat dengan menggunakan teknologi *Computer Vision*. Program ini menggunakan perangkat lunak sebagai berikut :

- a. OpenCV versi 2.1.0
- b. Microsoft Visual Studio 2010

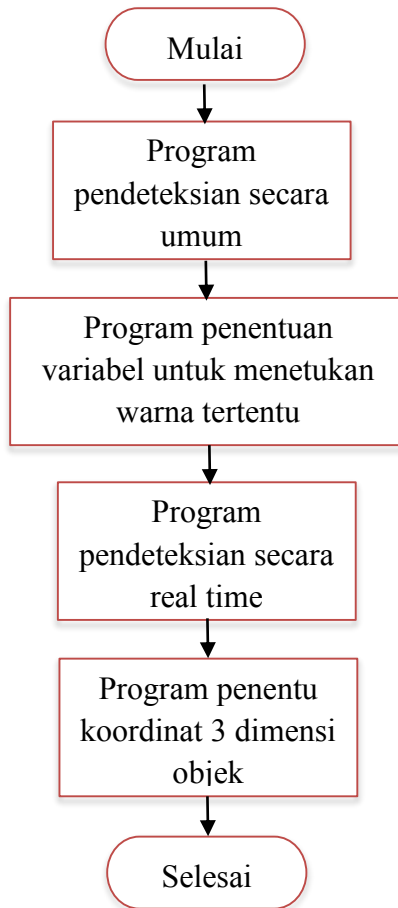
Pengujian dengan menggunakan beberapa data diperlukan untuk mengetahui performa dari program pendeteksi objek dan penentu koordinat 3 dimensi suatu objek tunggal ini. Pengujian ini dilakukan pada kondisi tertentu yaitu :

- a. Objek yang diambil berupa bola berdiameter ± 8 cm.
- b. Pengambilan objek dilakukan pada siang, sore, dan malam hari.

Sedangkan mesin pengolah yang digunakan dalam pengujian ini yaitu komputer dengan spesifikasi sebagai berikut:

- a. Sony VAIO E Series SVE141J11W
- b. *Processor Intel Core i3 2.4 GHz*
- c. *Memory RAM 2,00 GB*
- d. *System Type 64-Bit Operating system*
- e. *Operating system Windows 8*

Proses yang dikerjakan untuk program pendeteksian dan penentuan koordinat objek dapat dilihat pada diagram alir perancangan program Gambar 4.1.



Gambar 4.1 Diagram alir pembuatan program

Pada gambar 4.1 tersebut, secara umum proses perancangan program dibagi menjadi empat proses yaitu pembuatan program pendeteksian objek, program penentuan variabel, program pendeteksian secara *real-time*, dan program penentuan koordinat 3 dimensi objek. Setiap proses ini memiliki peranan masing-masing.

4.2. Konstruksi Program

Pada tahap konstruksi ini dibahas tentang tahap-tahap penerapan metode yang digunakan dalam pembuatan program pendeteksi dan penentu koordinat 3 dimensi objek tunggal. Proses konstruksi ini dimulai dari tahap deteksi objek, kemudian dilakukan penentuan jarak objek terhadap kamera serta penentuan koordinat x dan y objek terhadap kamera. Konstruksi program dibangun ke dalam kode program menggunakan *tools* yang telah ditentukan. Program yang digunakan untuk menerapkan metode tersebut kedalam kode program adalah Microsoft Visual Studio 2010 dengan *library* OpenCV dengan bahasa pemrograman C++.

4.2.1. Program pendeteksian objek

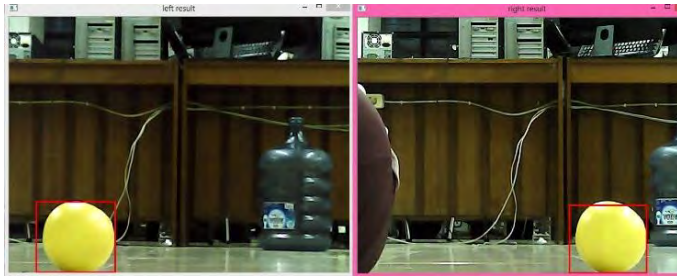
Kode program 1: *RGB to HSV*

```

IplImage *HSV=cvCreateImage(cvSize(frame-
>width,frame->height),8,3);
IplImage *HSV2=cvCreateImage(cvSize(frame2-
>width,frame2->height),8,3);
cvCvtColor(frame,HSV,CV_BGR2HSV);
cvCvtColor(frame2,HSV2,CV_BGR2HSV);

```

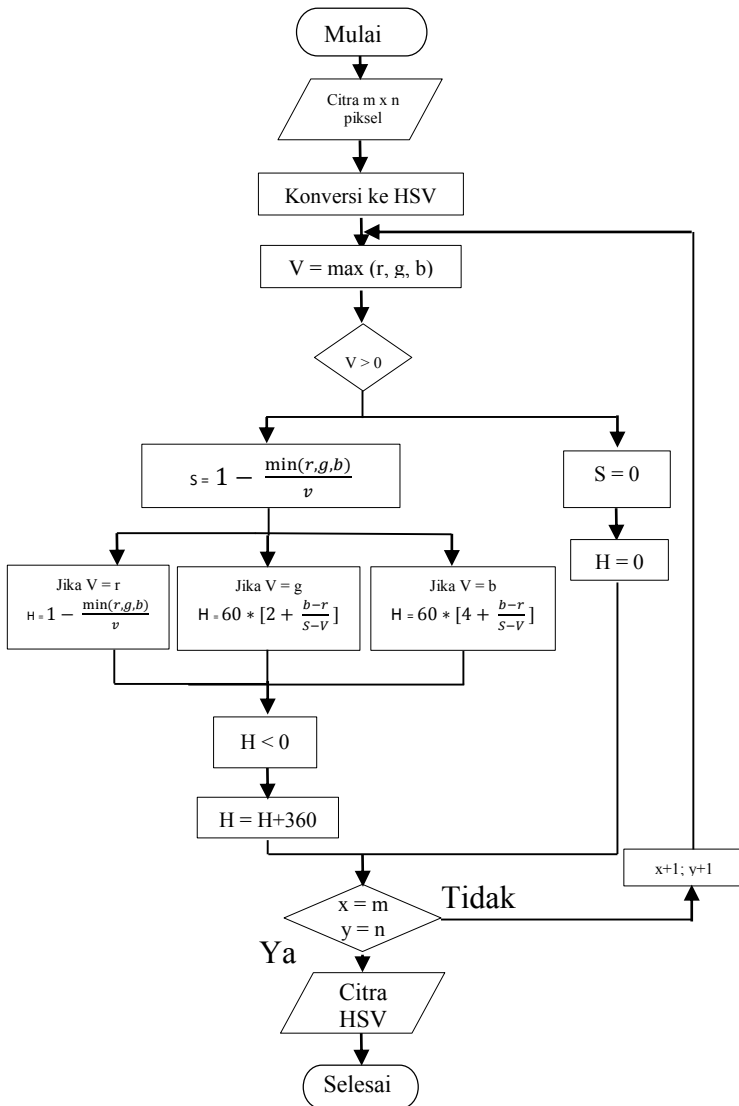
Pada proses pendeteksian objek proses yang dilakukan terlebih dahulu yaitu mengubah citra masukan RGB menjadi citra HSV yang dilakukan dengan menggunakan program. Hal ini dilakukan karena hasil *thresholding* yang diperoleh pada citra HSV akan lebih baik dibanding dengan hasil yang diperoleh dengan melakukan proses *thresholding* pada citra RGB.



Gambar 4.2 Citra *RGB* sebagai citra masukan frame kiri dan kanan

Frame hasil dari *capture* kamera USB berbentuk dalam format citra *RGB* (*Red Gren Blue*) dengan resolusi citra 640 x 480 *pixel*. Hasil dari *frame* tersebut akan diproses menjadi citra *HSV*.

Untuk merubah dari gambar *RGB* ke *HSV* secara manual dapat digunakan persamaan (3). Gambar 4.3 adalah diagram alir dari proses pengolahan citra *RGB* ke *HSV*, pada gambar tersebut dapat dilihat proses perubahan dari citra *RGB* menjadi citra *HSV*.



Gambar 4.3 Diagram alir proses pengolahan dari citra RGB ke HSV

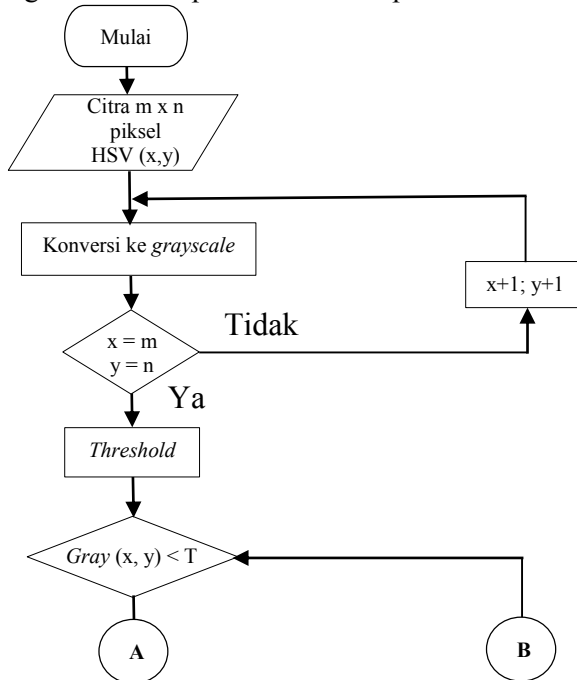
Kode program 2: *threshold*

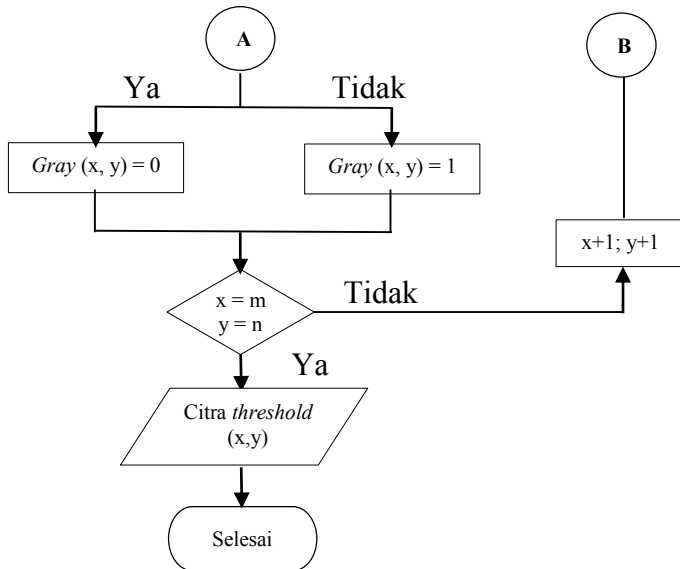
```

IplImage *thres=cvCreateImage(cvSize(frame-
>width,frame->height),8,1);
IplImage *thres2=cvCreateImage(cvSize(frame2-
>width,frame2->height),8,1);
cvInRangeS(HSV, cvScalar(hl,sl , vl), cvScalar(hh, sh,
vh), thres);
cvInRangeS(HSV2, cvScalar(hl,sl , vl), cvScalar(hh,
sh, vh), thres2);

```

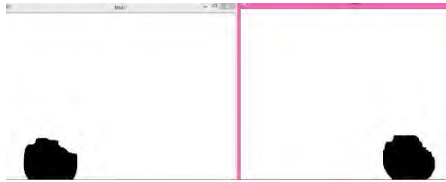
Kemudian dilakukan proses *threshold* pada citra hasil proses yang berupa citra HSV. Tipe *threshold* yang digunakan adalah *binary threshold* dengan *range* nilai HSV minimum (0, 0, 0) – maksimum (179, 255, 255 untuk HSV low dan 179, 255, 255 untuk HSV high). Berikut adalah diagram alir dari proses *threshold* pada citra HSV:



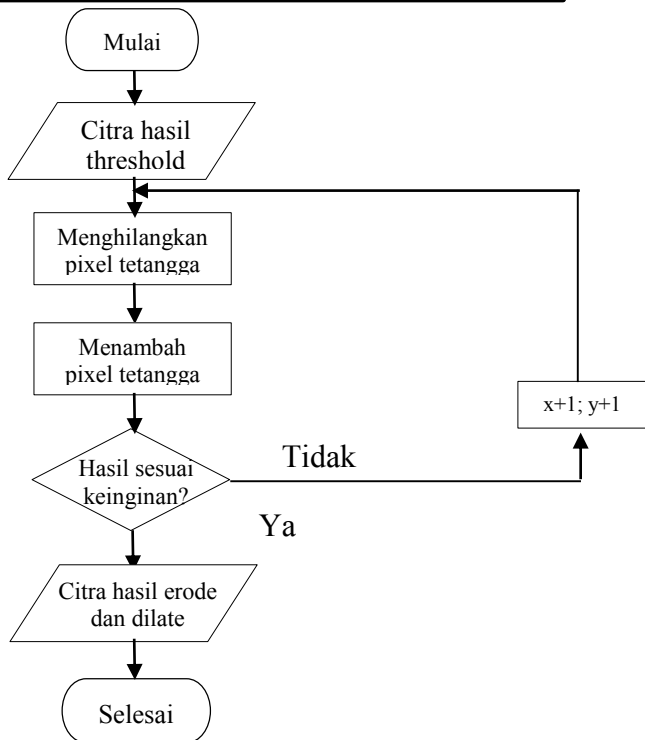


Gambar 4.4 Diagram alir proses *threshold*

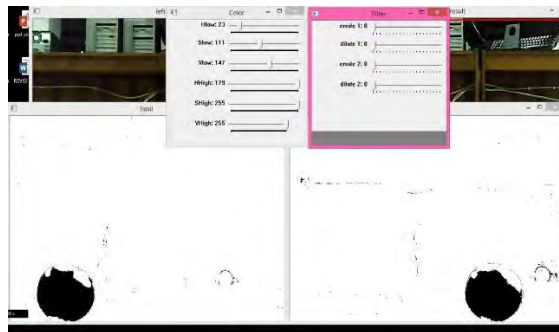
Hasil perubahan citra dari HSV ke citra biner yang dilakukan dengan tipe *binary threshold* terlihat seperti pada Gambar 4.5. pada citra tersebut terlihat bahwa hanya terdapat 2 warna yaitu hitam dan putih. Dimana warna hitam pada program ini menunjukkan objek dan warna putih merupakan latar belakang yang bukan merupakan bagian dari objek. Namun terkadang hasil yang diperoleh dari proses ini kurang sempurna, seperti yang terlihat pada gambar 4.5, pada gambar tersebut terlihat bahwa hasil *threshold* hanya sebagian dan kurang sempurna. Hal ini akan dibahas pada sub bab pengujian.

Gambar 4.5 Citra *threshold***Kode program 3: *erode dan dilate***

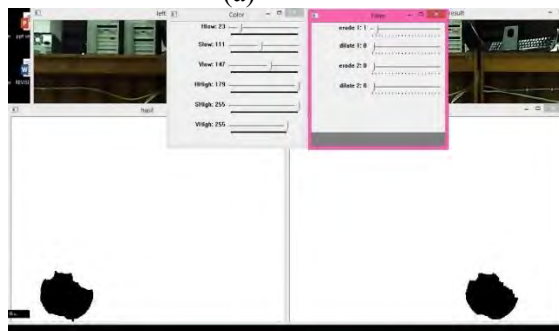
```
cvErode(thres,thres,element,ero1);
cvDilate(thres,thres,element,dil1);
cvErode(thres,thres,element,ero2);
cvDilate(thres,thres,element,dil2);
```

Gambar 4.6 Diagram alir proses *erode dan dilate*

Pada proses *threshold* ini diperlukan juga proses *noise filtering*. Dimana diagram alir dari proses ini dapat dilihat pada gambar 4.6. Hal ini disebabkan karena pada proses *threshold* biasanya terdapat gangguan/noise pada hasil citra yang diperoleh. *Noise filtering* bertujuan untuk membuang atau mengurangi gangguan tersebut agar citra *threshold* yang diperoleh sesuai dengan yang diinginkan. *Noise filtering* terdiri dari *erode* dan *dilate*, dimana *erode* bertujuan untuk mengurangi *pixel* sedangkan *dilate* bertujuan untuk menambah *pixel*.



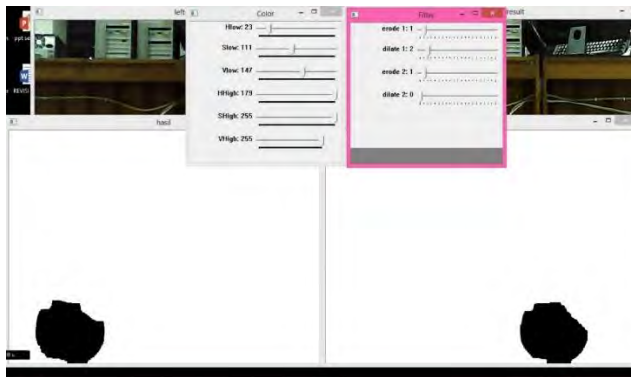
(a)



(b)

Gambar 4.7 (a) Citra *Threshold* sebelum dilakukan *erode* dan (b) Citra *Threshold* hasil *Erode 1 pixel* tetangga

Terlihat pada gambar 4.7 perbedaan Citra *threshold* sebelum dilakukan *noise filtering* dan setelah dilakukan proses. Pada citra 4.7 (a) terlihat masih terdapat *noise* yang dapat mengganggu pendeteksian objek. Citra *threshold* tersebut kemudian dilakukan *erode*. *Erode* dilakukan dengan pengecilan 1 *pixel* tetangga. Hasil *erode* terlihat seperti pada Gambar 4.7 (b). Citra *threshold* yang sudah di-*erode* kemudian dilakukan *dilate* dengan tiap 2 *pixel* tetangga dari *pixel* terluar. Hasil dari *dilate* tersebut terlihat seperti pada Gambar 4.8.



Gambar 4.8 Citra *Threshold* hasil *dilate* 2 *pixel* tetangga

Setelah semua proses tersebut dilakukan dilanjutkan dengan proses pendeteksian. Pendeteksian objek dilakukan dengan menggunakan cara pendeteksian warna. Setelah dilakukan proses *threshold* untuk mendapatkan citra biner, dimana objek akan berwarna hitam dan selain objek berwarna putih, maka koordinat *x* dan *y* dari titik tengah objek dapat ditemukan. Keberadaan objek juga dapat ditandai dengan penandaan lokasi menggunakan *rectangle*. Sehingga akan muncul kotak penanda berwarna merah disekeliling objek yang terdeteksi.

4.2.2 Pencarian Variabel *Range* Warna *Threshold*

Kode program 4: Mencari range warna threshold

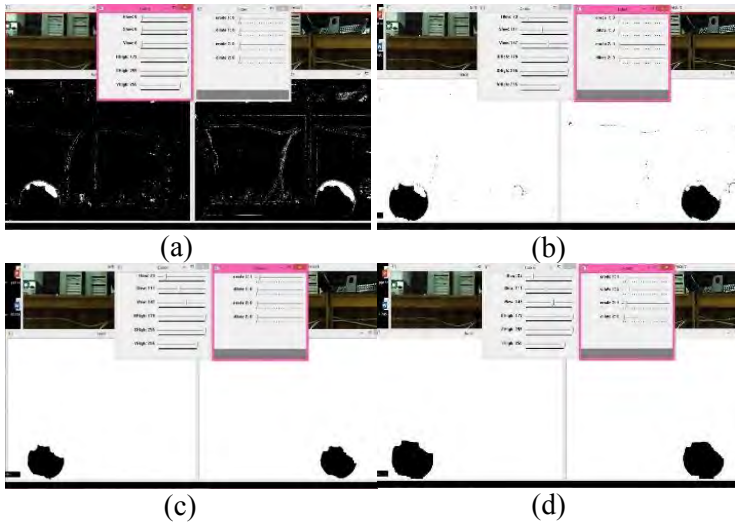
```
cvNamedWindow("Color",1);
cvCreateTrackbar("Hlow","Color", &hl,179,0);
cvCreateTrackbar("Slow","Color", &sl,255,0);
cvCreateTrackbar("Vlow","Color", &vl,255,0);
cvCreateTrackbar("HHhigh","Color", &hh,179,0);
cvCreateTrackbar("SHhigh","Color", &sh,255,0);
cvCreateTrackbar("VHhigh","Color", &vh,255,0);
```

Dari hasil citra HSV yang diperoleh kemudian dilakukan pencarian *range* citra tersebut untuk dilakukan perubahan ke citra *threshold* seperti pada Gambar 4.9.

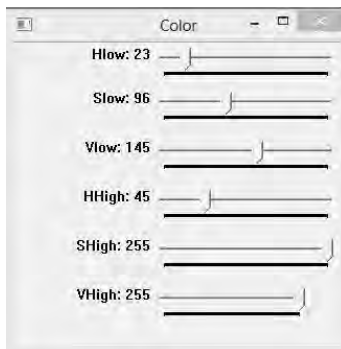


Gambar 4.9 *Trackbar* yang digunakan untuk mencari Perubahan Citra HSV ke Citra *Threshold*

Pada saat citra pertama kali akan dilakukan proses *threshold* akan tampak seperti pada gambar a, kemudian dilakukan pencarian *range* tersebut sehingga akan diperoleh perubahan *threshold* dengan menggunakan *trackbar* seperti pada gambar b dan dilakukan *noise filtering* sehingga akan terlihat seperti pada gambar c.



Gambar 4.10 Citra *Threshold* dari *Trackbar*



Gambar 4.11 Informasi Hasil Pencarian
Range dengan *trackbar*

Dari hasil pada gambar 4.11 dapat dilihat bahwa range yang dibutuhkan untuk mendeteksi warna kuning yaitu :

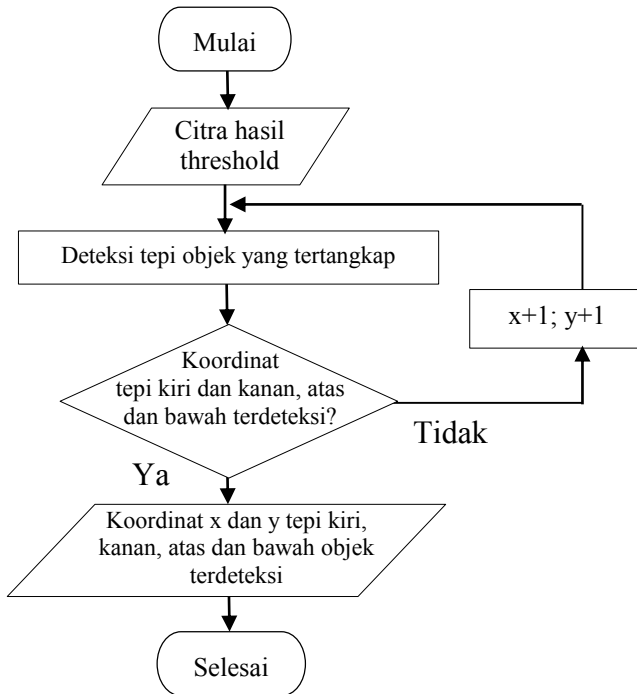
- Hlow = 23
- Slow = 96
- Vlow = 145
- Hhigh = 45
- Shigh = 255
- Vhigh = 255

Hasil dari *range* yang diperoleh kemudian digunakan untuk proses pendeteksian objek. Proses pendeteksian sangat bergantung pada hasil pengolahan citra yang sudah dilakukan, sehingga penentuan range ini harus dilakukan dengan beberapa percobaan untuk mendapatkan hasil yang terbaik. Kondisi lingkungan saat dilakukan juga sangat berpengaruh, sehingga nilai range yang diperoleh sewaktu – waktu bisa berubah sesuai kondisi pencahayaan ruangan tempat pengujian dilakukan.

4.2.3 Pencarian Variabel Pendeteksian Objek

Kode program 5: Mendeteksi Objek

```
for(int x=0;x<thres->width;x=x++)
for (int y=0;y<thres->height;y=y++)
{
    if(hasil->imageData[hasil-
>widthStep*y+x*hasil->nChannels]==0)
    {
        ukur.x= x;
    }
}
for (int y=0;y<thres->height;y=y++)
for(int x=0;x<thres->width;x=x++)
{
    if(hasil->imageData[hasil-
>widthStep*y+x*hasil->nChannels]==0)
    {
        ukur.y= y;
    }
}
```

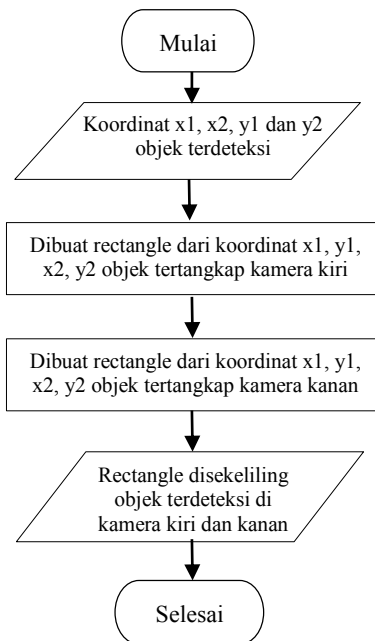


Gambar 4.12 Diagram alir pencarian koordinat tepi objek

Dalam pendeteksian objek tertentu melalui warna dapat dilakukan dengan memanfaatkan hasil dari proses threshold yang sudah dilakukan terlebih dahulu. Kemudian dari hasil tersebut koordinat dari tepi objek bisa ditemukan, dengan menggunakan program. Dari program tersebut nantinya akan diperoleh koordinat x dan y tepi kiri dan kanan objek, sehingga titik tengah dari objek tersebut bisa diketahui melalui perhitungan.

Kode program 6: Menunjukkan objek yang terdeteksi

```
cvRectangle( frame,cvPoint( ukur.x, ukur.y),cvPoint(
ukur1.x , ukur1.y),cvScalar( 0, 0, 255, 0 ), 2, 0, 0 );
cvRectangle( frame2,cvPoint( ukur2.x,
ukur2.y),cvPoint( ukur3.x , ukur3.y),cvScalar( 0, 0,
255, 0 ), 2, 0, 0 );
```



Gambar 4.13 Diagram alir pembuatan *rectangle*

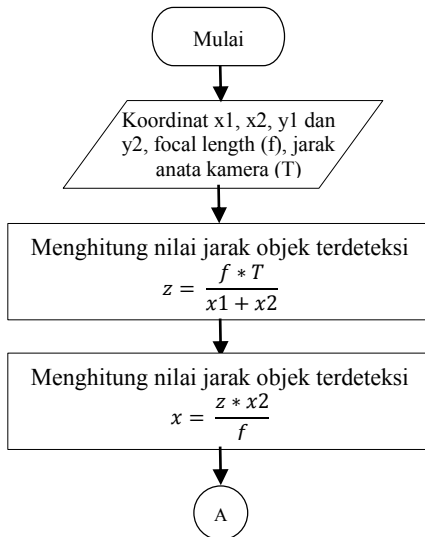
Setelah diperoleh titik koordinat tepi kiri dan kanan dari objek, dibuat program yang bertujuan untuk membuat persegi disekeliling objek. Hal itu bertujuan untuk menunjukkan lokasi dimana objek yang sudah terdeteksi. Sehingga pada saat program dijalankan maka

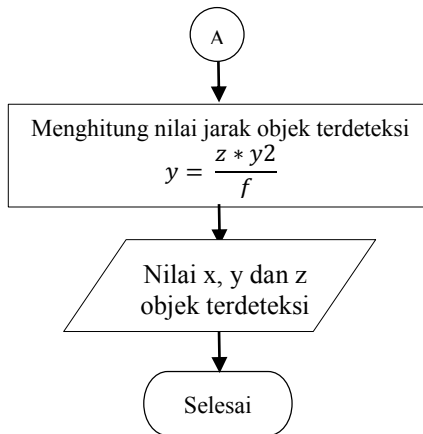
secara otomatis benda berwarna tertentu sesuai yang diinginkan terdeteksi dan secara otomatis akan ditandai dengan kotak merah disekeliling objek yang terdeteksi tersebut.

4.2.4 Program Penentuan Koordinat x, y, dan z

Kode program 7 : Menentukan koordinat z, x, dan y

```
int xfix=((ukur.x-ukur1.x)/2)+ukur1.x;
int yfix=((ukur.y-ukur1.y)/2)+ukur1.y;
int xfix2=(((ukur2.x-ukur3.x)/2)+ukur3.x);
int yfix2=(((ukur2.y-ukur3.y)/2)+ukur3.y);
int f=8.1876385326098966e+002;
int d=((640-xfix2)+xfix);
int s=abs(300*f)/d;
int z=s+((0.0179*(s*s))-(7.4910*s)+660.33);
int t=(xfix2*z)/f;
int x=(t-((1.5149*t)+42.429));
int u=(yfix2*z)/f;
int y=(u+((-1.4594*u)+46.061));
```





Gambar 4.14 Diagram alir penentuan nilai x, y, z objek terhadap kamera

Untuk mencari nilai koordinat z atau jarak objek terhadap kamera dipakai koordinat titik tengah dari objek yang terdeteksi di kamera kanan dan kamera kiri. Dari kedua koordinat titik tengah tersebut dibuat persamaan untuk menemukan nilai z. Persamaan yang digunakan yaitu persamaan (10). Sedangkan untuk mencari koordinat x dan y yang merupakan jarak vertikal dan horizontal objek terhadap kamera dapat digunakan persamaan (12) dan (14) pada bab 2.

Dari beberapa kali percobaan dilakukan untuk mengetahui hasil dari program pengukuran jarak objek terhadap kamera, diketahui bahwa hasil yang diperoleh tidak sesuai dengan jarak yang sebenarnya. Hal ini dapat disebabkan oleh banyak hal, seperti kualitas kamera yang kurang memadai, pencahayaan dan lain sebagainya. Hal yang sama juga terjadi pada saat mencari nilai x dan y. Nilai x dan y yang diperoleh dengan menggunakan program tidak sesuai dengan jarak yang sebenarnya. Untuk mengatasi hal tersebut dilakukan percobaan untuk

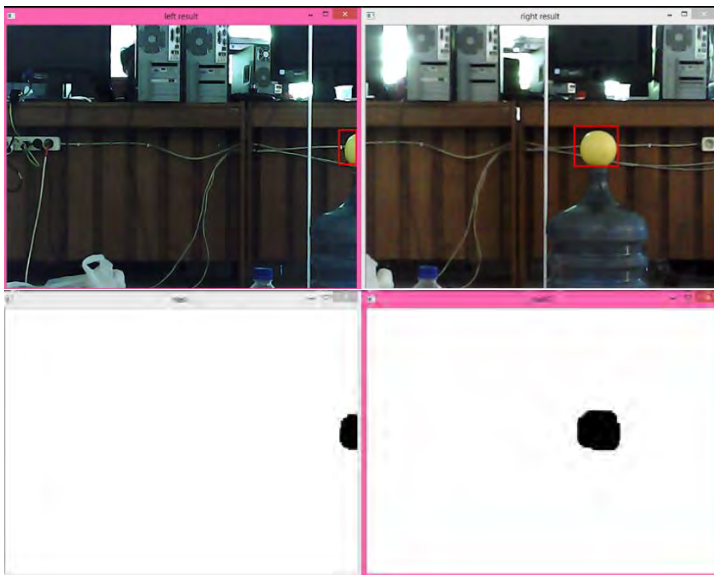
mengambil beberapa data yang bertujuan untuk mencari persamaan dari error hasil pengujian. Sehingga indikasi dari error tersebut dapat diketahui melalui suatu persamaan yang dihasilkan. Persamaan error tersebut akan dimasukkan kedalam persamaan sehingga hasil z , x dan y akan diperoleh sesuai atau mendekati dengan nilai yang sebenarnya.

BAB V

PENGUJIAN PENDETEKSIAN OBJEK

5.1 Pendeteksian objek

Pengujian dilakukan pada objek tunggal berwarna kuning dan berada pada posisi diam. Hasil yang diperoleh setelah beberapa kali pengujian dapat dilihat pada gambar 5.1

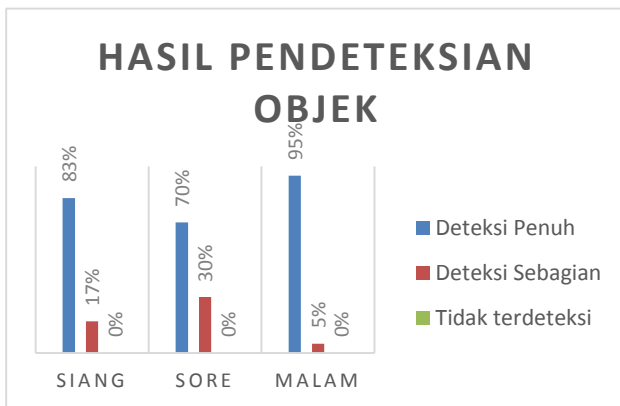


Gambar 5.1 Hasil pendeteksian objek

Pendeteksian tersebut dilakukan pada siang hari dan dilakukan pada kondisi pencahayaan dalam ruangan. Setelah dilakukan pengujian pada pagi, siang, maupun malam hari hasil yang diperoleh untuk pendeteksian ini sempurna. Kemampuan pendeteksian objek bola berwarna kuning 100% meskipun latar belakang dari objek tersebut terdiri dari berbagai warna. Namun hasil dari pendeteksian tersebut terbagi dalam dua kategori. Terdapat hasil pendeteksian objek penuh dan sebagian. Hasil

pendeteksian objek penuh merupakan hasil dari pendeteksian objek yang seluruh bentuk objek terdeteksi. Sedangkan pendeteksian objek sebagian yaitu merupakan hasil dari pendeteksian objek tidak sempurna atau hanya sebagian dari objek yang terdeteksi. Hal ini dapat disebabkan karena permukaan objek yang mengkilat. Sehingga pada saat cahaya mengenai objek ada sebagian dari permukaan objek tersebut yang memantulkan cahaya sehingga kamera menangkap hasil warna yang berbeda. Hasil warna yang ditangkap kamera menjadi putih sehingga dianggap bukan bagian dari objek. Persentase keakuratan program untuk pendeteksian objek bola kuning dapat dilihat pada grafik 4.16.

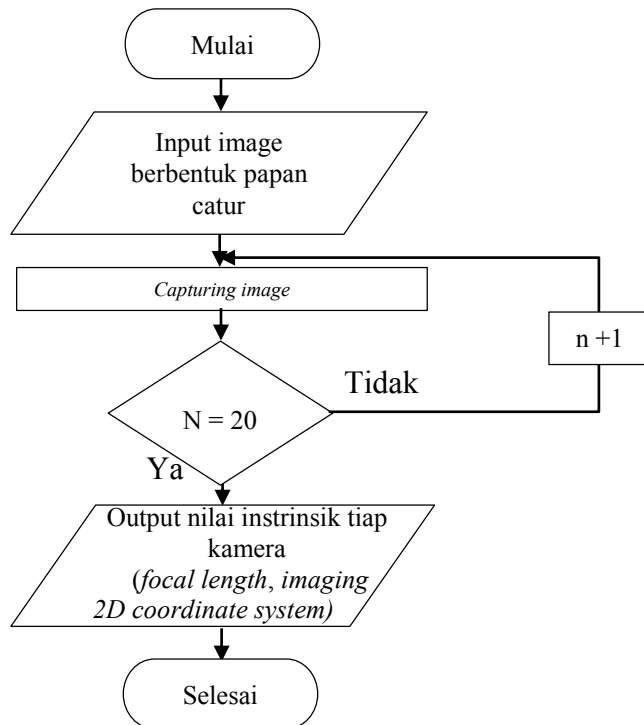
Dapat dilihat pada grafik tersebut bahwa persentase keakuratan pendeteksian pada siang dan sore hari lebih buruk dibanding dengan pengujian pada saat malam hari. Hal ini dikarenakan pada siang dan sore hari cahaya matahari yang masuk dalam ruangan terlalu banyak sehingga menyebabkan permukaan bola memantulkan cahaya. Akibatnya kamera menangkap warna yang berbeda sehingga program menganggap bagian tersebut bukan bagian dari objek. Hal ini terkadang masih terjadi meskipun pencahayaan dari ruangan sendiri dikurangi dengan cara mematikan lampu ruangan.



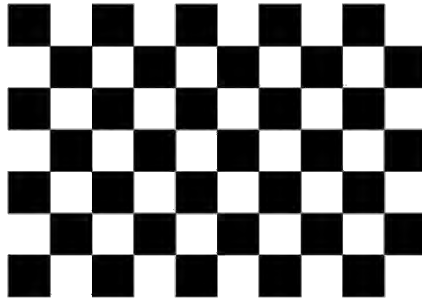
Gambar 5.2 Grafik hasil pendeteksian objek

5.2 Proses kalibrasi

Pada proses kalibrasi ini dilakukan yang dilakukan terlebih dahulu yaitu pembuatan program kalibrasi stereo kamera. Program ini sudah tersedia pada *library* OpenCV 2.1.0, hanya saja diperlukan sedikit penyesuaian agar program sesuai dengan data yang dipakai, seperti ukuran tiap kotak hitam dan putih papan catur yang dipakai untuk kalibrasi. Diagram alir yang digunakan untuk proses ini dapat dilihat pada gambar 4.15.



Gambar 5.3 Diagram alir proses kalibrasi



Gambar 5.4 Papan catur yang digunakan untuk proses kalibrasi



Gambar 5.5 Proses kalibrasi

Hasil yang diperoleh dari proses kalibrasi ini yaitu matriks M yang berisi nilai *focal length* dan sistem koordinat 2D. Matriks yang diperoleh yaitu sebagai berikut:

$$M1 = \begin{bmatrix} 8.1876385326098966 + 002 & 0 & 3.8793232252640030 + 002 \\ 0 & 8.1876385326098966 + 002 & 1.5817382135959343 + 002 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M2 = \begin{bmatrix} 8.1876385326098966 + 002 & 0 & 3.9115226159786960 + 002 \\ 0 & 8.1876385326098966 + 002 & 1.5417065737098014 + 002 \\ 0 & 0 & 1 \end{bmatrix}$$

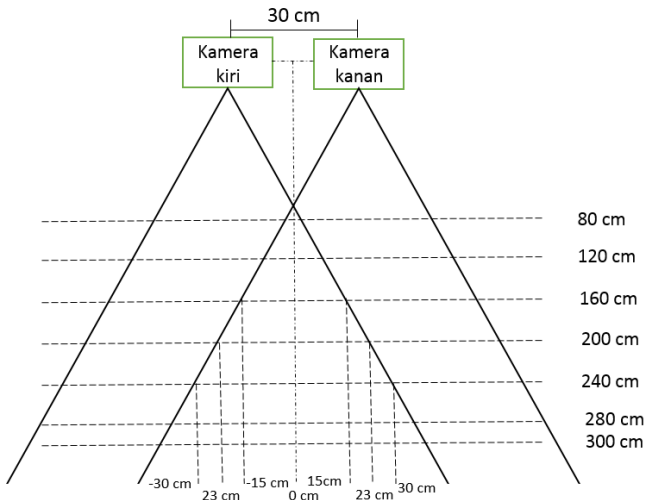
Dari matriks tersebut dapat diketahui nilai f , u_0 dan v_0 . Dimana nilai – nilai tersebut diketahui berdasarkan matriks M matriks sebagai berikut:

$$M = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Dimana f merupakan *focal length* dan u_0 dan v_0 merupakan sistem koordinat 2 dimensi. Data – data tersebut digunakan untuk proses penentuan nilai koordinat 3 dimensi dari objek yang menggunakan persamaan (10), (12), dan (14) pada bab II. Persamaan tersebut membutuhkan nilai *focal length* yang hanya dapat diketahui melalui proses kalibrasi ini.

5.3 Pendeteksian jarak (koordinat z)

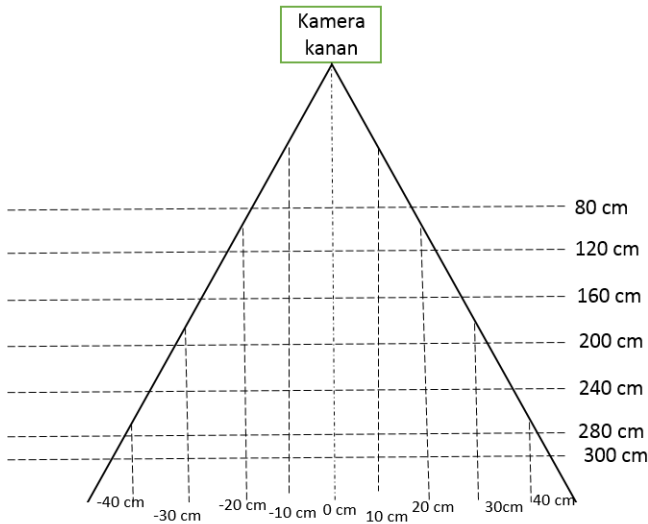
Pengujian dilakukan pada program pendeteksi jarak objek atau koordinat z. Pengujian dilakukan dengan menaruh objek di jarak tertentu yang sudah diketahui jarak yang sebenarnya. Skema pengambilan data dapat dilihat pada gambar 4.20 dan 4.21.



Gambar 5.6 Skema pengambilan data z dan x (tampak atas)



Gambar 5.7 Proses pengambilan data

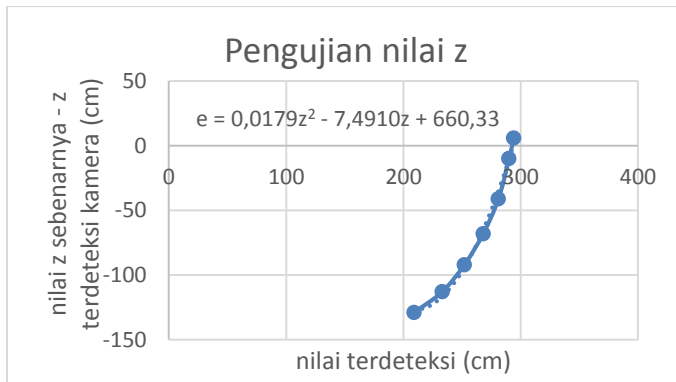


Gambar 5.8 Skema pengambilan data z dan y (tampak samping)

Pengambilan data z diambil pada jarak 80, 120, 160, 200, 240, 280 dan 300 cm. Selanjutnya untuk pengambilan data x dan y dapat dilihat pada gambar 4.20 dan 4.21 diambil pada jarak -30, -15, 0, 15, dan 30 cm untuk data x dan -40, -30, -20, -10, 0, 10, 20, 30, 40 cm untuk data y. Dari beberapa kali pengujian yang dilakukan diperoleh hasil yang dapat dilihat pada Tabel 1 pada lampiran.

Dapat dilihat dari tabel 1 pada lampiran bahwa nilai z yang diperoleh tidak sesuai dengan nilai z yang sebenarnya. Hal ini

dapat disebabkan karena perubahan posisi objek pada kondisi sebenarnya dan perubahan posisi objek yang ditangkap kamera tidak sama. Pada kondisi sebenarnya perubahan yang terjadi sangat besar, misalkan perubahan jarak dari 240 cm ke 300 cm, namun yang terdeteksi oleh kamera perubahan tersebut hanya 5 *pixel*. Untuk mengatasi hal tersebut diperlukan suatu persamaan garis yang dapat digunakan untuk mengurangi kesalahan pendeteksian nilai koordinat z. Aplikasi Microsoft Excel digunakan untuk menemukan persamaan tersebut, dengan memasukkan nilai error dan nilai rata – rata dari tiap pengujian maka akan diperoleh grafik seperti berikut :



Gambar 5.9 Grafik antara nilai error z dan nilai z terdeteksi kamera

Setelah grafik nilai error diperoleh, diketahui bahwa bentuk kurva dari nilai error tersebut cenderung berbentuk kurva polinomial. Sehingga diperoleh persamaan garis yang ditemukan dengan menggunakan pendekatan polinomial yaitu $e = 0,0179z^2 - 7,4910z + 660,33$. Nilai e tersebut merupakan nilai error dari nilai yang terdeteksi kamera. Sehingga nilai z yang diperoleh akan ditambah dengan nilai error tersebut. Dari hasil akhir yang sudah

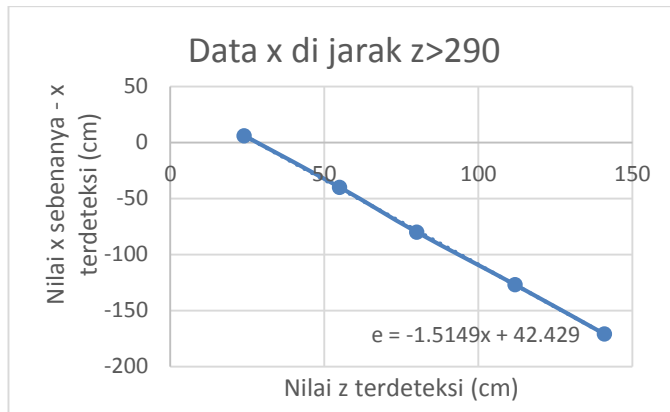
ditambah dengan nilai error, hasil nilai z dapat dilihat pada tabel 2 pada lampiran.

Dari hasil tersebut dapat diketahui bahwa keakuratan dan kepresisian dari program penentu nilai z cukup baik. Kesalahan dari hasil pengukuran yang diperoleh sebesar 0.0075%.

5.4 Pendeteksian jarak vertikal dan horizontal (koordinat x dan y)

Pengujian juga dilakukan pada program pendeteksi jarak vertical dan horizontal objek terhadap posisi kamera atau koordinat x dan y . Dari beberapa kali pengujian yang dilakukan diperoleh hasil yang dapat dilihat pada tabel 3.

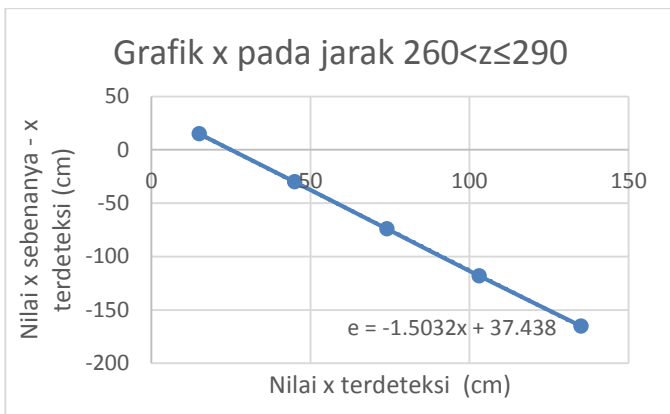
Dapat dilihat dari tabel 3 bahwa nilai x yang diperoleh kurang sesuai dengan nilai x yang sebenarnya juga. Faktor penyebab terjadinya kesalahan pengukuran ini juga sama seperti pada saat pengukuran koordinat z . Yaitu karena perubahan posisi objek pada kondisi sebenarnya dan perubahan posisi objek yang ditangkap kamera tidak sama. Solusi yang digunakan juga sama seperti metode sebelumnya. Namun terdapat perbedaan untuk mengurangi kesalahan pengukuran untuk nilai koordinat x ini. Hal ini dikarenakan pada saat dilakukan pengukuran nilai koordinat x , terdapat perbedaan seiring dengan perbedaan nilai koordinat z . Untuk mengatasi hal tersebut diperlukan suatu persamaan garis yang dapat digunakan untuk mengurangi kesalahan pendeteksian nilai koordinat x . Diperlukan persamaan garis di tiap jarak yang berbeda, untuk mempermudah pencarian nilai error yang akan dipakai digunakan *range* nilai. Aplikasi Microsoft Excel digunakan untuk menemukan persamaan tersebut dan dianalisa kembali dengan menggunakan aplikasi Minitab, dengan memasukkan nilai error dan nilai rata – rata dari tiap pengujian maka akan diperoleh grafik seperti berikut :



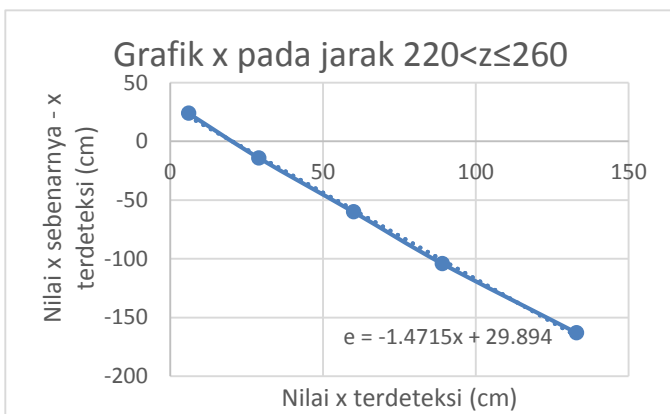
Gambar 5.10 Grafik antara nilai error dan nilai terdeteksi kamera jarak $z > 290$

Hal yang sama dilakukan juga pada data – data yang diambil di range jarak yang berbeda. Range yang digunakan yaitu $140 < z \leq 180$, $180 < z \leq 220$, $220 < z \leq 260$, $260 < z \leq 290$ dan $z > 290$. Untuk nilai error x pada jarak $z > 290$ diketahui persamaan garisnya yaitu $y = -1,5149x + 42,429$. Untuk grafik dari nilai error pada range selanjutnya dapat dilihat pada tabel 4 sampai 7 pada lampiran dan grafik dibawah.

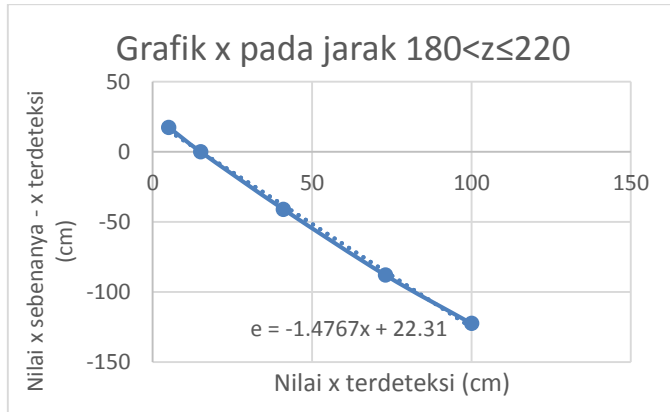
Dari data – data tersebut kemudian grafik dibuat untuk mencari persamaan garis. Persmaan garis ini yang akan digunakan untuk mengurangi kesalahan pengukuran. Grafik x untuk tiap – tiap range nilai z dapat dilihat pada grafik 4.25 sampai grafik 4.28.



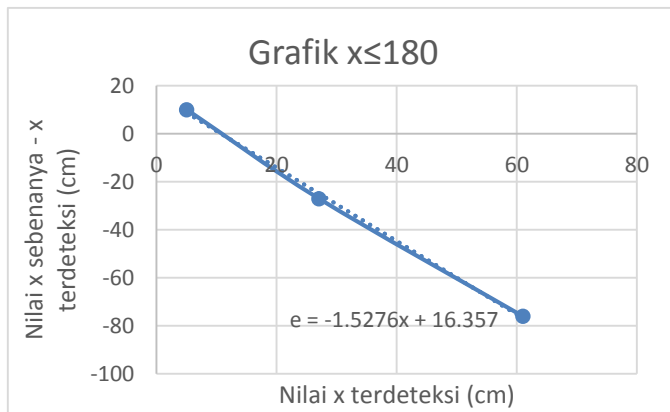
Gambar 5.11 Grafik antara nilai error x dan nilai x terdeteksi kamera pada jarak $260 < z \leq 290$



Gambar 5.12 Grafik antara nilai error x dan nilai x terdeteksi kamera pada jarak $220 < z \leq 260$



Gambar 5.13 Grafik antara nilai error x dan nilai x terdeteksi kamera pada jarak $180 < z \leq 220$



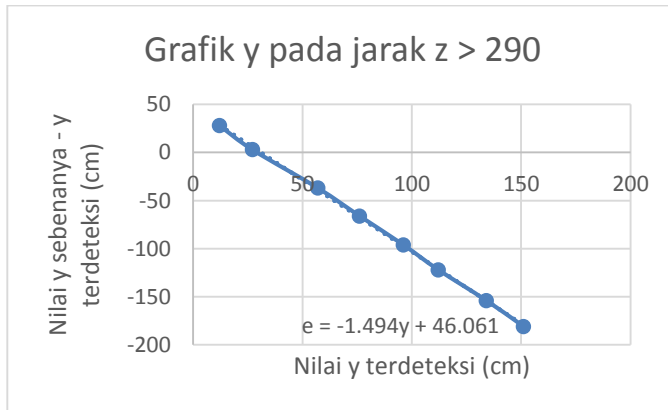
Gambar 5.14 Grafik antara nilai error x dan nilai x terdeteksi kamera pada jarak $z \leq 180$

Dari grafik yang dibuat berdasarkan data – data yang diperoleh saat dilakukan pengujian, diketahui kurva yang dihasilkan cenderung linear. Sehingga bisa didekati dengan persamaan garis

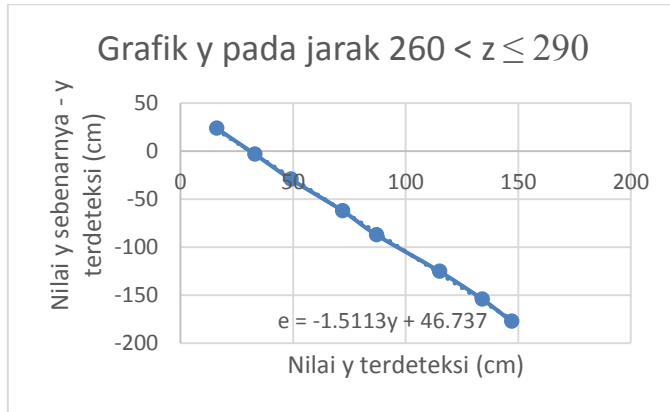
linear. Untuk jarak antara 260 dan 290 persamaan garis yang diperoleh yaitu $e = -1,5032x + 37,438$. Sedangkan untuk jarak z antara 220 cm dan 260 cm persamaan yang diperoleh yaitu $e = -1,4715x + 29,894$. Kemudian untuk z pada jarak antara 180 cm dan 220 cm diperoleh persamaan garis $e = -1,4767x + 22,31$. Dan untuk jarak dibawah 180 cm persamaan garis yang diperoleh $e = -1,5276x + 16,357$. Nilai error yang diperoleh dari persamaan – persamaan garis tersebut akan ditambahkan pada nilai x yang diperoleh oleh kamera.

Untuk mencari nilai kesalahan pada pengukuran nilai koordinat y dilakukan metode yang sama seperti pada pengukuran nilai koordinat x . Data – data yang diperoleh pada pengukuran nilai koordinat y dapat dilihat pada tabel 13 sampai 18 pada lampiran.

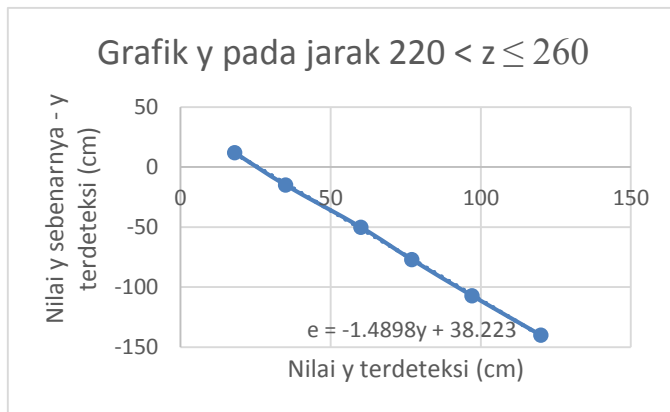
Dari data – data diatas kemudian grafik dibuat untuk mencari persamaan garis. Persmaan ini yang akan digunakan untuk mengurangi kesalahan pengukuran. Grafik y untuk tiap – tiap range nilai z dapat dilihat pada grafik 4.29 sampai grafik 4.34.



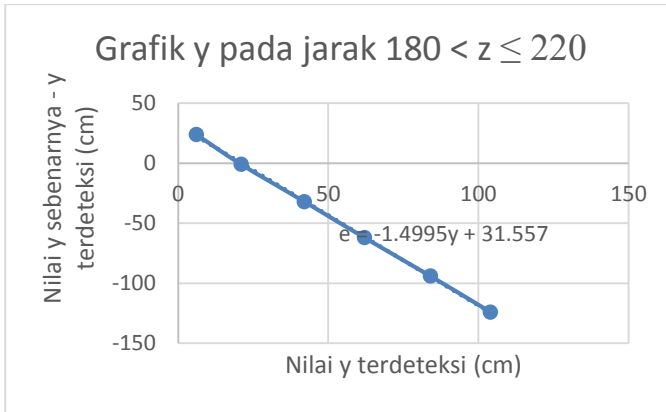
Gambar 5.15 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $z > 290$



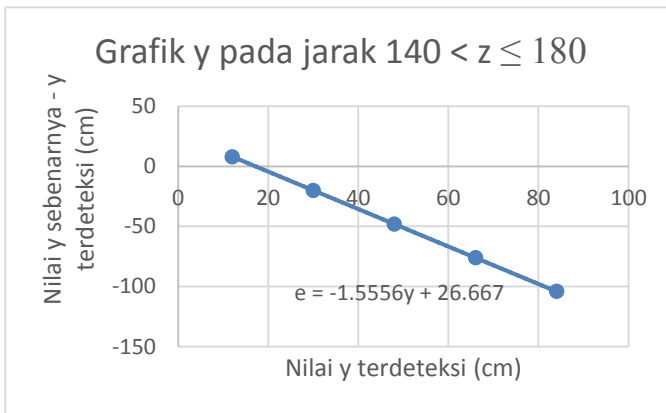
Gambar 5.16 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $260 < z \leq 290$



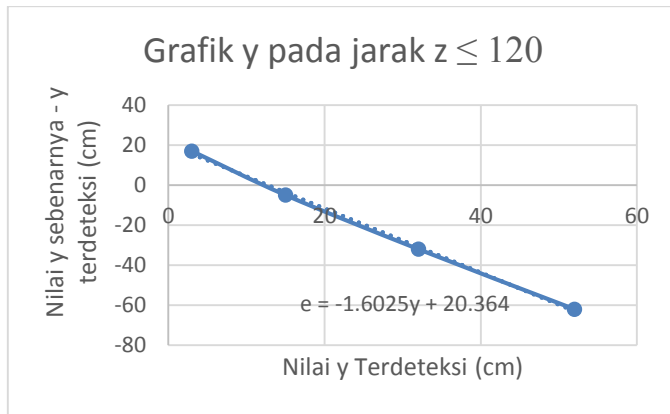
Gambar 5.17 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $220 < z \leq 260$



Gambar 5.18 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $180 < z \leq 220$



Gambar 5.19 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $140 < z \leq 180$



Gambar 5.20 Grafik antara nilai error y dan nilai y terdeteksi kamera pada jarak $z \leq 120$

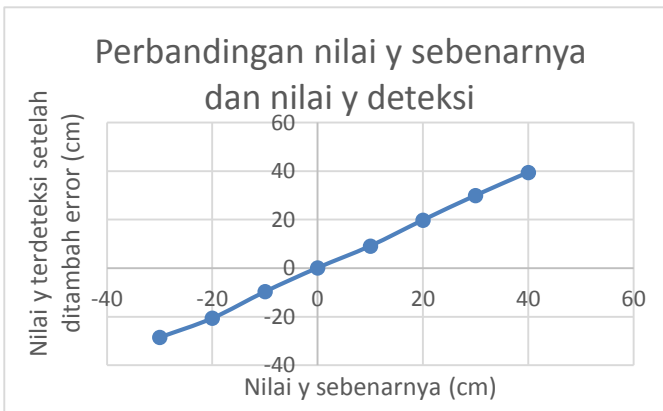
Dari grafik yang dibuat berdasarkan data – data yang diperoleh saat dilakukan pengujian, diketahui kurva yang dihasilkan cenderung linear. Sehingga bisa didekati dengan persamaan garis linear. Untuk jarak z diatas 290 cm persamaan garisnya yaitu $e = -1,494y + 46,061$. Untuk jarak antara 260 dan 290 persamaan garis yang diperoleh yaitu $e = -1,4898y + 38,223$. Sedangkan untuk jarak z antara 220 cm dan 260 cm persamaan yang diperoleh yaitu $e = -1,4715y + 29,894$. Kemudian untuk z pada jarak antara 180 cm dan 220 cm diperoleh persamaan garis $e = -1,4995y + 31,557$. Dan untuk jarak diantara 140 cm dan 180 cm persamaan garis yang diperoleh $e = -1,5556y + 26,667$. Pada jarak z dibawah 120 cm persamaan garisnya $e = -1,6025y + 20,364$. Nilai error yang diperoleh dari persamaan – persamaan garis tersebut akan ditambahkan pada nilai x yang diperoleh oleh kamera.

Setelah persamaan – persamaan tersebut dimasukkan dalam program nilai error pada pengukuran nilai koordinat x dan y semakin kecil. Data hasil penjumlahan error dapat dilihat pada

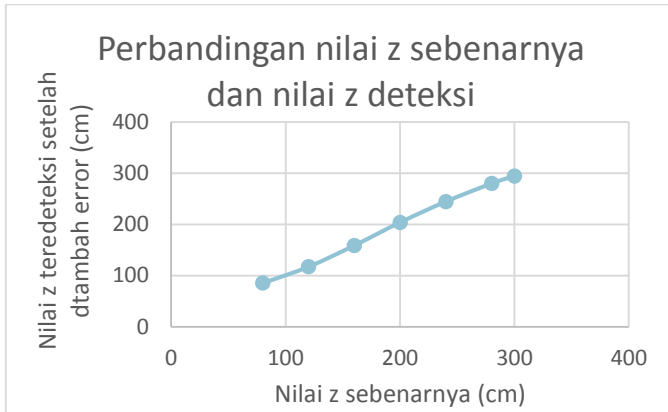
tabel 19 sampai 24 pada lampiran. Dari data tersebut dapat dihitung nilai error untuk pengukuran nilai koordinat x dan y. Untuk pengukuran nilai koordinat x memiliki error sebesar 0.092% dan untuk pengukuran nilai koordinat y memiliki error sebesar 0.085%.



Gambar 5.21 Grafik perbandingan nilai x sebenarnya dan nilai x terdeteksi



Gambar 5.22 Grafik perbandingan nilai y sebenarnya dan nilai y terdeteksi



Gambar 5.23 Grafik perbandingan nilai z sebenarnya dan nilai z terdeteksi

(Halaman ini sengaja dikosongkan)

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Beberapa kesimpulan yang didapat pada penelitian tugas akhir ini adalah sebagai berikut:

1. Proses kalibrasi stereo kamera dilakukan dengan menggunakan papan catur yang berukuran 2,6 cm x 2,6 cm. Proses ini menghasilkan matriks yang berisi *focal length* sebesar $8.1876385326098966e + 002$ mm baik untuk kamera kiri maupun kanan.
2. Pendeteksian objek tunggal menunjukkan hasil yang baik dengan hasil pendeteksian sempurna yang didapat yaitu masing – masing 83%, 70% dan 95% untuk kondisi siang, sore dan malam. Jarak terjauh yang berhasil dideteksi secara real-time yaitu sejauh 500 cm dan jarak terdekat agar objek dapat terdeteksi yaitu 80 cm. Namun jika untuk menentukan koordinat 3 dimensi dari objek program hanya mampu menentukan koordinat objek maksimal sejauh 300 cm dan minimal jarak yang terdeteksi yaitu 80 cm. Pendeteksian nilai koordinat z atau jarak objek terhadap kamera pada program juga menunjukkan hasil yang baik yaitu dengan nilai kesalahan yang didapat sebesar 0,0075%. Pendeteksian nilai koordinat x dan y atau jarak vertikal dan horizontal objek terhadap kamera pada program menunjukkan hasil yang baik yaitu dengan nilai kesalahan yang didapat sebesar 0,092% dan 0,085%. Faktor yang mempengaruhi ketidaksempurnaan pendeteksian adalah adanya bagian dari objek bola yang memantulkan cahaya sehingga kamera tidak mendeteksi bagian tersebut sebagai objek dan ketidaksempurnaan persamaan

yang digunakan untuk mencari error karena hanya dilakukan dengan pendekatan.

6.2 Saran

Berdasarkan dari hasil pengujian dan pembahasan pada tugas akhir ini, terdapat beberapa saran pengembangan penelitian pendeteksian objek tunggal secara real-time dan penentuan koordinat 3 dimensi adalah sebagai berikut :

1. Pendeteksian objek sebaiknya menggunakan kamera dengan resolusi tinggi karena akan sangat mempengaruhi hasil pendeteksian.
2. Area tangkap dari kamera dengan jarak antar kamera yang hanya 30 cm masih dianggap kurang. Sehingga sebaiknya jika ingin menambah luasan area tangkap, posisi kamera tidak menghadap lurus kearah depan, tetapi sedikit miring kearah luar. Penggunaan kamera dengan *wide lens* dan pengurangan jarak antar kedua kamera juga bisa digunakan sebagai alternatif agar area tangkap dari kamera lebih luas.

LAMPIRAN

Lampiran Hasil Pengujian Pendeteksian Objek

Tabel 1. Hasil pengujian penentuan koordinat z

Jarak sebenarnya (cm)	I	II	III	IV	V	Rata - rata	Selisih
80	209	210	209	209	210	209	-129
120	233	232	233	232	233	233	-113
160	252	252	253	250	251	252	-92
200	268	268	267	268	268	268	-68
240	281	282	281	280	282	281	-41
280	290	290	290	289	291	290	-10
300	294	295	294	293	295	294	6

Tabel 2. Hasil pengujian z setelah ditambah nilai error

jarak sebenarnya (cm)	I	II	III	Rata - rata	Selisih
80	81	84	83	83	3
120	118	120	119	119	-1
160	161	160	159	160	0
200	200	201	200	200	1
240	247	244	246	246	6
280	283	280	281	281	1
300	303	300	299	301	1

Tabel 3. Hasil pengujian x untuk nilai z diatas 290

Nilai Sebenarnya (cm)	I	II	III	IV	V	Rata - rata	Selisih
-30	141	140	142	139	138	141	-171

-15	112	113	114	110	109	112	-127
0	80	79	80	81	80	80	-80
15	55	55	53	56	55	55	-40
30	24	23	24	25	24	24	6

Tabel 4. Hasil pengujian x untuk nilai $260 < z \leq 290$

Nilai Sebenarnya (cm)	I	II	III	IV	V	Rata - rata	Selisih
30	15	16	17	14	14	15	15
15	45	44	43	47	45	45	-30
0	77	74	74	74	70	74	-74
-15	103	103	103	104	104	103	-118
-30	133	135	133	136	136	135	-165

Tabel 5. Hasil pengujian x untuk nilai $220 < z \leq 260$

Nilai Sebenarnya (cm)	I	II	III	IV	V	Rata - rata	Selisih
30	6	8	6	5	4	6	24
15	29	30	28	27	29	29	-14
0	58	60	60	59	62	60	-60
-15	90	88	89	90	89	89	-104
-30	133	134	132	133	133	133	-163

Tabel 6. Hasil pengujian x untuk nilai $180 < z \leq 220$

Nilai Sebenarnya (cm)	I	II	III	IV	V	Rata - rata	Selisih
22.5	5	6	4	5	5	5	18
15	15	30	28	27	29	26	-11
0	40	41	43	42	40	41	-41

-15	73	75	72	73	73	73	-88
-22.5	100	100	102	98	100	100	-123

Tabel 7. Hasil pengujian x untuk nilai $140 < z \leq 180$

Nilai Sebenarnya (cm)	I	II	III	IV	V	Rata - rata	Selisih
15	5	6	4	5	5	5	10
0	28	27	28	27	26	27	-27
-15	61	60	63	62	60	61	-76

Tabel 8. Hasil pengujian x untuk nilai z diatas 290 setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
-30	-31	-30	-30	-30	0
-15	-16	-14	-15	-15	0
0	1	0	1	1	1
15	15	13	14	14	-1
30	31	30	30	30	0

Tabel 9. Hasil pengujian x untuk nilai $260 < z \leq 290$ setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
30	30	32	30	31	-1
15	15	15	14	15	0
0	0	2	1	1	-1
-15	-14	-15	-15	-15	0
-30	-30	-31	-31	-31	-1

Tabel 10. Hasil pengujian x untuk nilai $220 < z \leq 260$ setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
30	27	28	25	27	3
15	16	17	15	16	-1
0	2	3	1	2	-2
-15	-12	-15	-9	-12	-3
-30	-33	-30	-35	-33	3

Tabel 11. Hasil pengujian x untuk nilai $180 < z \leq 220$ setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
23	20	22	18	20	3
15	15	16	15	15	0
0	3	3	3	3	-3
-15	-12	-14	-11	-12	-3
23	-25	-24	-26	-25	3

Tabel 12. Hasil pengujian x untuk nilai $z \leq 180$ setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
15	14	15	14	14	1
0	2	3	2	2	-2
-15	-16	-17	-15	-16	1

Tabel 13 Hasil pengujian y untuk nilai z diatas 290

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
40	12	13	11	12	28
30	26	27	28	27	3
20	57	55	59	57	-37
10	78	74	76	76	-66
0	96	95	96	96	-96
-10	112	112	112	112	-122
-20	134	136	132	134	-154
-30	151	151	150	151	-181

Tabel 14 Hasil pengujian y untuk nilai $260 < z \leq 290$

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
40	16	15	16	16	24
30	34	33	35	34	-4
20	47	45	49	47	-27
10	79	79	80	79	-69
0	89	87	85	87	-87
-10	115	116	114	115	-125
-20	134	134	133	134	-154
-30	149	147	145	147	-177

Tabel 15 Hasil pengujian y untuk nilai $220 < z \leq 260$

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
30	16	20	18	18	12
20	35	34	36	35	-15
10	60	61	60	60	-50
0	79	75	77	77	-77
-10	97	96	98	97	-107
-20	119	120	121	120	-140

Tabel 16 Hasil pengujian y untuk nilai $180 < z \leq 220$

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
30	7	8	6	6	24
20	21	22	21	21	-1
10	40	42	44	42	-32
0	61	62	63	62	-62
-10	84	84	83	84	-94
-20	104	105	104	104	-124

Tabel 17 Hasil pengujian y untuk nilai $140 < z \leq 180$

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
20	11	10	12	12	8
10	30	31	30	30	-20
0	46	48	47	48	-48
-10	66	65	67	66	-76
-20	84	84	83	84	-104

Tabel 18 Hasil pengujian y untuk nilai $z \leq 140$

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
20	2	3	3	3	17
10	15	14	16	15	-5
0	31	32	33	32	-32
-10	52	53	52	52	-62

Tabel 19. Hasil pengujian y untuk nilai z diatas 290 setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
40	40	41	40	40	0
30	33	30	35	33	3
20	18	18	17	18	2
10	9	8	9	9	1
0	-1	0	-1	-1	1
-10	-9	-10	-9	-9	1
-20	-20	-22	-18	-20	0
-30	-29	-29	-30	-29	1

Tabel 20. Hasil pengujian y untuk nilai $260 < z \leq 290$ setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
40	39	38	39	39	-1
30	30	33	28	30	0
20	23	23	22	23	3
10	7	8	7	7	-3
0	3	0	3	2	3

-10	-12	-10	-13	-12	-2
-20	-21	-23	-20	-21	-1
-30	-28	-30	-26	-28	2

Tabel 21. Hasil pengujian y untuk nilai $220 < z \leq 260$ setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
30	28	27	28	28	-2
20	20	20	20	20	0
10	8	9	7	8	-2
0	0	0	1	0	0
-10	-10	-10	9	-4	0
-20	-22	-21	-23	-22	-2

Tabel 22. Hasil pengujian y untuk nilai $180 < z \leq 220$ setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
30	29	29	30	29	-1
20	21	20	21	21	1
10	11	10	12	11	1
0	1	0	1	1	1
-10	-10	-12	-8	-10	0
-20	-20	-20	-20	-20	0

Tabel 23. Hasil pengujian y untuk nilai $140 < z \leq 180$ setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
20	20	21	20	20	0
10	11	11	10	11	1
0	0	0	0	0	0
-10	-10	-11	-9	-10	0
-20	-20	-22	-18	-20	0

Tabel 24. Hasil pengujian y untuk nilai $140 < z \leq 180$ setelah ditambah nilai error

Nilai Sebenarnya (cm)	I	II	III	Rata - rata	Selisih
20	17	18	16	17	-3
10	9	9	10	9	-1
0	-1	0	-1	-1	-1
-10	-12	-14	-13	-13	-3

Lampiran Rancang Bangun Program

Berikut ini adalah *coding* pembangun program Program Pendeteksi, Pelacak dan Penentu Koordinat 3 Dimensi Objek Tunggal

```
#include "cv.h"
#include "highgui.h"
#include "math.h"
#include <cv.h>
#include <highgui.h>
#include <math.h>
#include <cxtypes.h>
#include <iomanip>
#include <stdio.h>
```

```

#include <conio.h>
#include "tserial.h"
#include <string.h>
#define PI 3.14159265

char dataserial[8];
CvPoint mousep[7];
CvPoint statpos[17];
CvPoint MouseFilter(CvPoint mop);
CvPoint ukur, ukur1, ukur2, ukur3, tengah=cvPoint(0,0);
int sta_test(CvPoint pos);

IplConvKernel *element;
IplConvKernel *element2;
void GetDesktopResolution(int& horizontal, int& vertical);
CvPoint pt=cvPoint(0,0);
int add_pt=0;

void on_mouse( int event, int x, int y, int flags, void*
param )
{
    if( event == CV_EVENT_MOUSEMOVE )
    {
        pt = cvPoint(x,y);
        add_pt = 1;
    }
}

int main()
{
    int x=0,y=0;
    int xs=0, ys=0, z=0;
    double param, teta1, teta2;
    param = 1.0;
    int hl=23,sl=96,vl=145,hh=45,sh=255,vh=255;
    CvCapture* capture = 0;;
    IplImage *frame;
    int data=0,H=0,S=0,V=0;
    capture = cvCaptureFromCAM(1);
    cvNamedWindow("left result",1);

```

```

int ero1=1,ero2=0,dil1=2,dil2=0;
int h1r=23,s1r=96,v1r=145,hhr=45,shr=255,vhr=255;
CvCapture* capture2 = 0;;
IplImage *frame2;
int data2=0,Hr=0,Sr=0,Vr=0;
capture2 = cvCaptureFromCAM(2);
cvNamedWindow("right result",1);
int ero1r=1,ero2r=0,dil1r=2,dil2r=0;
int k,j;

//=====
cvNamedWindow("Color",1);
cvCreateTrackbar("Hlow","Color", &h1,179,0);
cvCreateTrackbar("Slow","Color", &s1,255,0);
cvCreateTrackbar("Vlow","Color", &v1,255,0);
cvCreateTrackbar("HHhigh","Color", &hh,179,0);
cvCreateTrackbar("SHhigh","Color", &sh,255,0);
cvCreateTrackbar("VHhigh","Color", &vh,255,0);
//=====
cvNamedWindow("Filter",1);
cvCreateTrackbar("erode 1","Filter", &ero1,20,0);
cvCreateTrackbar("dilate 1","Filter", &dil1,20,0);
cvCreateTrackbar("erode 2","Filter", &ero2,20,0);
cvCreateTrackbar("dilate 2","Filter", &dil2,20,0);
//=====
while(1)
{
    frame = cvRetrieveFrame(capture);
    frame2 = cvRetrieveFrame(capture2);
    cvMirror(frame,frame,90);
    cvMirror(frame2,frame2,90);
    IplImage
*thres=cvCreateImage(cvSize(frame->width,frame-
>height),8,1);
    IplImage
*thres2=cvCreateImage(cvSize(frame2->width,frame2-
>height),8,1);
    IplImage
*HSV=cvCreateImage(cvSize(frame->width,frame->height),8,3);
    IplImage
*HSV2=cvCreateImage(cvSize(frame2->width,frame2-
>height),8,3);

```



```

                                IplImage
*hasil=cvCreateImage(cvSize(frame->width,frame-
>height),8,1);

                                IplImage
*hasil2=cvCreateImage(cvSize(frame2->width,frame2-
>height),8,1);

                                cvCvtColor(frame,HSV,CV_BGR2HSV);
                                cvCvtColor(frame2,HSV2,CV_BGR2HSV);
                                cvInRangeS(HSV, cvScalar(hl,s1 , v1),
cvScalar(hh, sh, vh), thres);
                                cvInRangeS(HSV2, cvScalar(hl,s1 ,
v1), cvScalar(hh, sh, vh), thres2);
                                element =
cvCreateStructuringElementEx (9,9,4,4, CV_SHAPE_RECT,NULL);
                                cvErode(thres,thres,element,ero1);
                                cvDilate(thres,thres,element,dil1);
                                cvErode(thres,thres,element,ero2);
                                cvDilate(thres,thres,element,dil2);
                                cvErode(thres2,thres2,element,ero1);
                                cvDilate(thres2,thres2,element,dil1);
                                cvErode(thres2,thres2,element,ero2);
                                cvDilate(thres2,thres2,element,dil2);

//=====PROSESTHRESHOLDING=====
                                for(int x=0;x<hasil->width;x=x++)
                                for (int y=0;y<hasil->height;y=y++)
                                {
                                        if (thres->imageData[thres-
>widthStep*y+x*thres->nChannels]==0)
                                                hasil->imageData[hasil-
>widthStep*y+x*hasil->nChannels]=255;
                                        else
                                                hasil->imageData[hasil-
>widthStep*y+x*hasil->nChannels]=0;
                                }
                                for(int x1=0;x1<hasil2-
>width;x1=x1++)
                                for (int y1=0;y1<hasil2-
>height;y1=y1++)
                                {
                                        if (thres2->imageData[thres2-
>widthStep*y1+x1*thres2->nChannels]==0)

```

```

        hasil2->imageData[hasil2-
>widthStep*y1+x1*hasil2->nChannels]=255;
        else
            hasil2->imageData[hasil2-
>widthStep*y1+x1*hasil2->nChannels]=0;
    }

    //=====MENDETEKSI OBJEK=====//
    for(int x=0;x<thres->width;x=x++)
    for (int y=0;y<thres->height;y=y++)
    {
        if(hasil->imageData[hasil-
>widthStep*y+x*hasil->nChannels]==0)
        {
            ukur.x= x;
        }
    }

    for(int x=0;x<thres2->width;x=x++)
    for (int y=0;y<thres2->height;y=y++)
    {
        if(hasil2->imageData[hasil2-
>widthStep*y+x*hasil2->nChannels]==0)
        {
            ukur2.x= x;
        }
    }

    for (int y=0;y<thres->height;y=y++)
    for(int x=0;x<thres->width;x=x++)
    {
        if(hasil->imageData[hasil-
>widthStep*y+x*hasil->nChannels]==0)
        {
            ukur.y= y;
        }
    }

    for (int y=0;y<thres2->height;y=y++)
    for(int x=0;x<thres2->width;x=x++)
    {
        if(hasil2->imageData[hasil2-
>widthStep*y+x*hasil2->nChannels]==0)

```

```

        {
            ukur2.y= y;
        }
    }

    for(int x=thres->width-1;0<x;x=x--)
    for (int y=thres->height-1;0<y;y=y--)

    {
        if(hasil->imageData[hasil-
>widthStep*y+x*hasil->nChannels]==0)
        {
            ukur1.x= x;
        }
    }

    for(int x=thres2->width-1;0<x;x=x--)
    for (int y=thres2->height-1;0<y;y=y-
-)
    {
        if(hasil2->imageData[hasil2-
>widthStep*y+x*hasil2->nChannels]==0)
        {
            ukur3.x= x;
        }
    }

    for (int y=thres->height-1;0<y;y=y--)

    for(int x=thres->width-1;0<x;x=x--)
    {
        if(hasil->imageData[hasil-
>widthStep*y+x*hasil->nChannels]==0)
        {
            ukur1.y= y;
        }
    }

    for (int y=thres2->height-1;0<y;y=y-
-)
    for(int x=thres2->width-1;0<x;x=x--)
    {

```

```

        if(hasil2->imageData[hasil2-
>widthStep*y+x*hasil2->nChannels]==0)
        {
            ukur3.y= y;
        }
    }

    //=====rectangledisekelilingobjek=====
    cvRectangle( frame,cvPoint( ukur.x,
ukur.y),cvPoint( ukur1.x , ukur1.y),cvScalar( 0, 0, 255,
0 ), 2, 0, 0 );
    cvRectangle( frame2,cvPoint( ukur2.x,
ukur2.y),cvPoint( ukur3.x , ukur3.y),cvScalar( 0, 0, 255,
0 ), 2, 0, 0 );

    //=====MENCARI NILAI X,Y, dan
Z=====//
    int xfix=((ukur.x-
ukur1.x)/2)+ukur1.x;
    int yfix=((ukur.y-
ukur1.y)/2)+ukur1.y;
    int xfix2=((ukur2.x-
ukur3.x)/2)+ukur3.x);
    int yfix2=((ukur2.y-
ukur3.y)/2)+ukur3.y);
    int f=8.1876385326098966e+002;
    int d=((640-xfix2)+xfix);;
    int s=abs(300*f)/d;
    int z=s+((0.0179*(s*s))-
(7.4910*s)+660.33);
    int t=(xfix2*z)/f;

    if (int (z>=140) && (z<180))
    {
        x =t+(-1.5276*t)+16.357;
    }

    else if ((z>=180)&&(z<220))
    {
        x =t+(-1.4767*t)+22.31;
    }

```

```

else if ((z>=220)&&(z<260))
{
    x =t+(-1.4715*t)+29.894;
}
else if ((z>=260)&&(z<290))
{
    x =t+(-1.5032*t)+ 37.438;

}
else if ((z>=290))
{
    x =t+(-1.5149*t)+42.429;

}

int u=(yfix2*z)/f;

if (int (z>=80) && (z<140))
{
    y =u+(-1.6025*u)+ 20.364;

}

else if ((z>=140)&&(z<180))
{
    y =u+(-1.5556*u)+ 26.667;

}

else if ((z>=180)&&(z<220))
{
    y =u+(-1.4995*u)+ 31.557;

}

else if ((z>=220)&&(z<260))
{
    y =u+((-1.4898*u)+ 38.223);

}

else if ((z>=260)&&(z<290))
{
    y =u+(-1.5113*u)+ 46.737;

```

```

    }
    else if ((z>=290))
    {
        y = u+((-1.494*u) + 46.061);
    }
    //=====Perubahan dari radian ke
derajat=====//
    xs = (1*x)+(0*y)+(0*z)+0;
    //xs= k+((-2.111*k)-0.0019);
    ys = (0*x)+(1*y)+(0*z)-14;
    //ys= j-((-0.8714*j)+12.242);
    teta1 = (atan ((double)ys/(double)z)
* (180/PI)) + 90;
    teta2 = atan ((double)xs/(double)z)
* (180/PI) +90;

    //=====//

    system("cls");
    printf("x1 : %d , y1 : %d
\n",xfix,yfix);
    printf("xr : %d , yr : %d
\n",xfix2,yfix2);
    //printf("x (pixel): %d \n",((xfix-
xfix2)/2)+xfix2);
    //printf("y (pixel): %d \n",((yfix-
yfix2)/2)+yfix2);

    printf("x : %d \n",x);
    printf("t : %d \n",t);
    printf("u : %d \n",u);
    printf("y : %d \n",y);
    printf("s : %d \n",s);
    printf("z : %d \n",z);
    printf("x senjata : %d \n",xs);
    printf("y senjata : %d \n",ys);
    printf("teta1 : %f \n", teta1);
    printf("teta2 : %f \n", teta2);

    cvShowImage("hasil",hasil);
    cvShowImage("left result",frame);
    cvShowImage("hasil2",hasil2);
    cvShowImage("right result",frame2);

```

```
        if(cvWaitKey(1) == 'q')
            break;

        cvReleaseImage( &thres);
        cvReleaseImage( &thres2);
        cvReleaseImage( &HSV);
        cvReleaseImage( &HSV2);
        cvReleaseImage( &hasil);
        cvReleaseImage( &hasil2);
    }
    cvDestroyWindow("Filter");
    cvDestroyWindow("Color");
    cvReleaseImage( &frame);
    cvReleaseImage( &frame2);
    cvReleaseCapture( &capture);
    cvReleaseCapture( &capture2);
    cvDestroyWindow("left result");
    cvDestroyWindow("right result");
    return 0;
}
```

DAFTAR PUSTAKA

- [1] Kurniawan, Y. (2008).” Implementasi Ultrasonik Level Detektor Pada Sistem Monitoring Tangki PENDAM Pada SPBU”.
- [2] Kuncorojati, A. (2015). “Rancang Bangun Pelontar Peluru Yang Dilengkapi Dengan Kamera Stereo Untuk Pendektesian Target Secara Otomatis”.
- [3] Imron, M.A., (2015). “Pendeteksian dan Pelacakan 2D Multi Objek Secara Real – time dengan Menggunakan Kamera Tunggal”.
- [4] Rusjdi, D., Kusumo, C.J., dkk. (2012). “Perancangan Simulasi Penentuan Koordinat Objek Di Dalam Ruang Menggunakan Kamera Digital”.
- [5] Finali, A. (2015). “Pengukuran Jarak Menggunakan Metode *Stereo Vision* Untuk Mengidentifikasi Objek Bergerak”.
- [6] Tjandrasa, H. (2009). “Stereo Vision Untuk Pengukuran Jarak Objek Dengan Mendeteksi Tepi Subimage”
- [7] Ji, Q., & Zhu, Z. (2007). Novel Eye Gaze Tracking Techniques under Natural Head Movements. *IEEE* .
- [8] Adri, M. (2009). ”*Computer Vision : Basic Concept of Computer Vision*”.
- [9] Putra, D. (2010). “*Pengolahan Citra Digital*”. Yogyakarta. ANDI.
- [10] Gonzalez, R. C., Woods R. E. (2002), *Digital Image Processing, Second ed.* New Jersey: Prentice-Hall.

- [11] Putra, D. (2010). *“Pengolahan Citra Digital”*, Yogyakarta, Andi Offset.
- [12] R. D. Kusumanto. (2011). *“Klasifikasi Warna Menggunakan Pengolahan Warna HSV”*
- [13] Representasi Model Warna RGB Menggunakan HSL dan HSV. (2011). Dipetik pada 1 November 2015, dari <https://mhstekkomp.wordpress.com/2011/05/07/representasi-model-warna-rgb-menggunakan-hsl-dan-hsv/>
- [15] OpenCV. (2011).Dipetik pada 30 Oktober 2015, dari <https://www.willowgarage.com/pages/software/opencv>
- [16] Sistem Koordinat Kartesius. (2015). Dipetik pada 22 Oktober 2015, dari https://id.m.wikipedia.org/wiki/Sistem_koordinat_Kartesius
- [17] Ogura, T., Mizuchi, Y., dkk. (2013). *“Distance Measurement System using Stereo Camera for Automatic Ship Control”*. Tokyo University.
- [18] Brahmbhatt, S. (2013). *“Practical OpenCV, Hands On Project For Computer Vision On The Windows, Linux and Raspberry PI Platform”*.

BIODATA PENULIS



Wardah Choirina Lutfi lahir di Surabaya pada 9 Februari 1993, merupakan anak kedua dari tiga bersaudara. Penulis telah menempuh pendidikan formal yaitu TK Khadijah Surabaya (1997-1999), SD Khadijah 1 Surabaya (1999-2005), SMP Negeri 19 Surabaya (2005-2008), SMA Negeri 2 Surabaya (2008-2011). Penulis melanjutkan pendidikan kuliah dengan mengikuti SNMPTN Jalur Undangan dan diterima di Jurusan Teknik Mesin, Institut Teknologi Sepuluh Nopember. Penulis terdaftar dengan NRP 2111100025.

Penulis mengambil Bidang Studi Manufaktur di Laboratorium Perancangan dan Pengembangan Produk Jurusan Teknik Mesin. Penulis berkegiatan dalam bidang akademik diantaranya menjadi grader praktikum Pengukuran Teknik dan mata kuliah Program Komputer.

Dalam bidang organisasi, penulis pernah menjadi Bendahara Umum di Himpunan Mahasiswa Mesin (2013-2014). Sebelumnya penulis menjadi staff di Departemen Kesejahteraan Mahasiswa Himpunan Mahasiswa Mesin (2012-2013). Penulis juga aktif mengikuti berbagai macam seminar dan pelatihan untuk meningkatkan *softskill*. Untuk semua informasi dan masukan dapat menghubungi penulis melalui email lutfiwardah@gmail.com.